SARIK: Uma proposta de *framework* para o aprimoramento da segurança em Kubernetes por meio de políticas de rede

Jonathan G. P. dos Santos[1], Geraldo P. Rocha Filho[2] (Coorientador), Vinícius P. Gonçalves[1] (Orientador)

[1]Universidade de Brasília (UnB)

[2]Universidade Estadual do Sudoeste da Bahia (UESB)

jonathanti@unb.br, geraldo.rocha@uesb.edu.br, vpgvinicius@unb.br

Resumo. *A adoção crescente do Kubernetes como plataforma de orquestração de contêineres traz benefícios para a gestão de aplicações distribuídas, mas também desafios de segurança, especialmente quanto ao controle de tráfego entre componentes. Esta dissertação apresenta o framework SARIK (Segurança Automática de Regras de Iptables no Kubernetes), um framework que automatiza políticas de rede para aprimorar a segurança de clusters Kubernetes. A metodologia inclui a integração do SARIK ao kube-proxy para aplicar, de forma dinâmica, regras de bloqueio e controle de tráfego. Em um ambiente de teste com Minikube, Prometheus e Grafana, foram avaliadas métricas como latência, taxa de resposta e taxa de transmissão em diferentes cenários de tráfego. Os resultados indicam que o SARIK aprimora a segurança ao reduzir vulnerabilidades de rede, mantendo o desempenho do cluster praticamente inalterado. A análise indica que o SARIK representa um avanço em segurança automatizada para Kubernetes, equilibrando proteção e eficiência operacional, com potencial para futuras adaptações em larga escala.*

Abstract. *The increasing adoption of Kubernetes as a container orchestration platform brings benefits to the management of distributed applications but also poses security challenges, especially concerning the control of traffic between components. This dissertation presents the SARIK framework (Automatic Security of Iptables Rules in Kubernetes), a framework that automates network policies to enhance the security of Kubernetes clusters. The methodology includes the integration of SARIK into kube-proxy to dynamically apply blocking rules and traffic control. In a test environment with Minikube, Prometheus, and Grafana, metrics such as latency, response rate, and throughput were evaluated across different traffic scenarios. The results indicate that SARIK improves security by reducing network vulnerabilities while keeping the cluster's performance practically unchanged. The analysis indicates that SARIK represents an advancement in automated security for Kubernetes, balancing protection and operational efficiency, with the potential for future large-scale adaptations.*

1. Introdução

A popularização do Kubernetes como plataforma de orquestração de contêineres trouxe avanços significativos para o gerenciamento de aplicações distribuídas, oferecendo escalabilidade e flexibilidade em ambientes complexos [Calixto 2024]. No entanto, essa

popularidade também revela desafios significativos de segurança, especialmente no que se refere ao controle de tráfego de rede entre contêineres e a proteção contra acessos não autorizados. Em um contexto de *clusters* Kubernetes que abrigam múltiplas aplicações e, frequentemente, dispositivos de *Internet of Things* (IoT), torna-se essencial garantir que a comunicação entre contêineres ocorra de maneira controlada e segura, minimizando as vulnerabilidades que possam comprometer a integridade do sistema.

No domínio da segurança de contêineres, diversas abordagens têm sido propostas para melhorar o isolamento de tráfego e mitigar vulnerabilidades de rede. O *framework* Tocker, desenvolvido [Balabanian and Henriques 2019], com foco em ambientes Docker, onde restringe a comunicação entre contêineres ao bloquear portas desnecessárias e introduz *firewall*. Essa solução foi uma das pioneiras ao abordar o problema do isolamento de tráfego em contêineres e inspirou pesquisas subsequentes, incluindo o desenvolvimento do SARIK no contexto de Kubernetes. Outra solução relevante é o BASTION [Nam et al. 2020], que propõe uma pilha de rede de segurança para contêineres, proporcionando isolamento eficaz e mitigação de ataques, com ganhos de até 25,4% no desempenho da rede.

Trabalhos recentes também exploram monitoramento e detecção de intrusão em Kubernetes. [Kulathunga 2021] propôs um *Intrusion Detection System* (IDS) dinâmico, enquanto [Levy Rocha et al. 2023] utilizaram aprendizado de máquina para identificar anomalias em ambientes de contêineres. Outras contribuições, como o Sysdig *Secure* [Secure 2023] e o Kub-Sec [Zhu and Gehrmann 2022], atendem a necessidades específicas de conformidade e automação de políticas de segurança, ampliando o controle de acesso e monitoramento de rede. Embora essas soluções ofereçam avanços em áreas específicas, a plataforma Kubernetes tem um conjunto de desafios de segurança, especialmente pela sua arquitetura distribuída e altamente dinâmica. Entre os principais desafios estão a ausência de políticas de rede padrão restritivas — por padrão, todos os Pods podem se comunicar entre si —, a falta de isolamento efetivo entre *namespaces*, a exposição indevida de APIs e serviços sensíveis, e a dificuldade de aplicar controles de acesso refinados em nível de rede. Além disso, o gerenciamento manual de regras e políticas, combinado à rápida variedade do ambiente (com Pods sendo criados e destruídos constantemente), contribui para a complexidade de manter a segurança de forma contínua e precisa. Portanto, ambientes Kubernetes, especialmente aqueles que lidam com dispositivos IoT, demandam políticas de segurança sólidas e dinâmicas para lidar com a crescente diversidade de dispositivos e fluxos de tráfego, além de mitigarem possíveis ameaças internas e externas.

Para enfrentar esses desafios, são necessárias políticas de rede automatizadas e eficientes, que consigam controlar o tráfego entre contêineres e nós de forma precisa e com o mínimo impacto na performance do sistema. Nesse contexto, o *framework* SARIK [dos Santos et al. 2025] surge como uma solução para a aplicação automatizada de políticas de rede, visando aprimorar a segurança em *clusters* Kubernetes sem sacrificar o desempenho operacional.

O objetivo desta dissertação é apresentar o SARIK e demonstrar como ele pode preencher lacunas críticas na segurança de ambientes Kubernetes. O *framework* utiliza integração com o *kube-proxy* para aplicar políticas de rede automaticamente, controlando o tráfego entre os contêineres por meio da inserção dinâmica de regras de bloqueio e filtragem. Essa abordagem permite ao SARIK fornecer uma camada de proteção adicional contra possíveis ataques e vulnerabilidades, sem exigir alterações manuais contínuas.

As principais contribuições desta dissertação são:

1. Propor e desenvolver o *Framework* SARIK para a automação de políticas de rede em Kubernetes, melhorando a segurança de forma prática e escalável.
2. Integração de regras de rede dinâmicas com o *kube-proxy*, proporcionando um controle eficiente do tráfego de entrada e saída em contêineres.
3. Avaliação do desempenho do SARIK em ambientes simulados, demonstrando que a solução aprimora a segurança sem comprometer significativamente a performance do sistema.

As pesquisas realizadas nesta dissertação resultaram em contribuições para a comunidade acadêmica e profissional, culminando em duas publicações. A primeira, intitulada "SARIK - *Framework* para automatizar a segurança em ambientes de orquestração Kubernetes[1]", foi apresentada no Salão de Ferramentas do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (Qualis A4) em maio de 2022. Este trabalho introduziu a versão inicial do SARIK, focando na automação de segurança por meio de regras de *iptables* aplicadas diretamente nos *Pods*. A segunda publicação, "*Enhancing IoT device security in Kubernetes: An approach with network policies and the SARIK framework*[2]", foi publicada na revista "*Future Generation Computer Systems*" da Elsevier (Qualis A1) em janeiro de 2025. Este artigo representa a principal contribuição da dissertação, aprimorando o SARIK com integração ao *kube-proxy* para controle dinâmico do tráfego de rede, oferecendo uma abordagem sólida para a segurança em Kubernetes.

Tabela 1. Resumo das publicações.

Artigo	Publicado	Local	Qualis	Fator de impacto
[dos Santos et al. 2025]	Periódico	*Future Generation Computer Systems*	A1	7.2
[dos Santos et al. 2022]	Conferência	SBRC	A4	–

2. *Framework* SARIK

O *framework* SARIK foi desenvolvido para aprimorar a segurança em Kubernetes por meio de uma abordagem automatizada e modular, focada em implementar políticas de rede de forma eficiente para proteger *clusters* contra ameaças e acessos não autorizados. O acrônimo SARIK significa "**S**egurança **A**utomática de **R**egras de **I**ptables no **K**ubernetes", refletindo seu objetivo central de automatizar o controle de tráfego em *clusters* Kubernetes utilizando regras configuradas dinamicamente. A modularidade do SARIK permite uma integração flexível com o ecossistema Kubernetes, facilitando a adaptação e a expansão para atender diferentes requisitos de segurança e desempenho. A seguir, detalham-se os componentes principais da arquitetura do SARIK, sua integração com o *kube-proxy* para gerenciamento de políticas de rede e os desafios enfrentados em sua implementação.

2.1. Arquitetura e componentes

A arquitetura é composta por quatro componentes, conforme observado na Figura 1, que trabalham em conjunto para implementar uma solução de segurança:

[1]Artigo SBRC 2022: `https://doi.org/10.5753/sbrc_estendido.2022.223438`
[2]Artigo Elsevier 2025: `https://doi.org/10.1016/j.future.2024.107485`

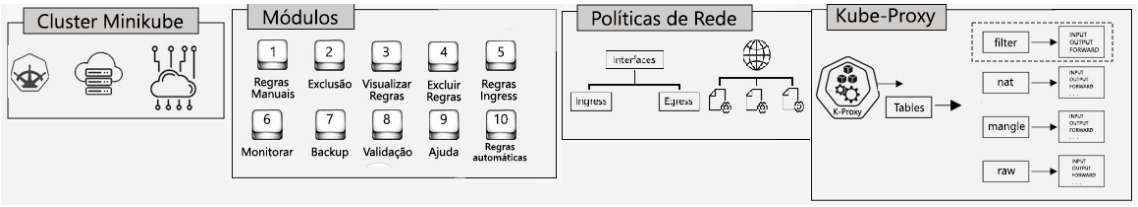

Figura 1. Arquitetura do framework SARIK

1. *Cluster* Minikube/Kubernetes: O SARIK é implementado em um ambiente Kubernetes utilizando o Minikube, uma versão local simplificada que facilita o desenvolvimento e teste. O uso do Minikube permite a integração e validação das funcionalidades do SARIK em um ambiente controlado, sendo ideal para experimentação e ajustes iniciais.
2. Módulos do *framework* SARIK: O SARIK é projetado de maneira modular, facilitando a automação e o gerenciamento de políticas de rede. Cada módulo desempenha uma função específica, são no total 10 módulos.
3. Políticas de rede: O SARIK automatiza a criação e aplicação de políticas de rede em *YAML*, ajustando as regras dinamicamente com base nas necessidades de segurança identificadas. Essas políticas controlam o tráfego de saída e entrada nos *Pods*, restringindo acessos não autorizados e mitigando ameaças.
4. *Kube-Proxy*: O *Kube-proxy* atua como intermediário na aplicação das políticas de rede, traduzindo os manifestos *YAML* gerados pelo SARIK em regras de *iptables*. Essa integração permite um controle eficiente do tráfego e a aplicação rápida de regras de bloqueio e filtragem de pacotes.

O SARIK é composto por módulos que operam em conjunto para monitorar, gerenciar e aplicar políticas de segurança. O principal componente do SARIK é o módulo de políticas de rede, que se integra ao *kube-proxy* para atuar diretamente sobre o fluxo de tráfego nos *clusters* Kubernetes. Essa integração permite que o SARIK manipule regras de *iptables* e aplique filtros de rede automaticamente, facilitando a automação e o gerenciamento de políticas de rede. Cada módulo desempenha uma função específica:

- Configuração manual e automática de políticas: Permite ao usuário definir manualmente as políticas de rede tanto para o tráfego de saída (*egress*) quanto para o de entrada (*ingress*), além de contar com um módulo para geração automática dessas políticas.
- Exclusão e visualização de políticas: Inclui módulos para exclusão automática e manual das políticas aplicadas, bem como um módulo de visualização que exibe as regras de rede ativas.
- Monitoramento e *backup*: O SARIK oferece um módulo de monitoramento em tempo real das políticas de rede e um módulo de *backup* para garantir que as regras possam ser restauradas rapidamente em caso de falhas.
- Validação de políticas: Este módulo verifica a conformidade das regras aplicadas com as melhores práticas de segurança, identificando possíveis vulnerabilidades.

2.2. Desafios e limitações

A implementação do SARIK encontrou alguns desafios, principalmente devido ao uso de contêineres privilegiados para manipulação de regras de *iptables* no ambiente Kubernetes.

Inicialmente, essa abordagem exigia que os pods fossem criados com a *flag privileged*, o que aumentava a exposição do sistema a riscos de segurança. Para mitigar essa limitação, o SARIK foi aprimorado para gerenciar as regras de bloqueio em camadas de rede e transporte diretamente no *kube-proxy*, reduzindo a necessidade de permissões elevadas e, assim, aumentando a segurança do ambiente.

Outro desafio foi garantir a compatibilidade com diferentes configurações de Kubernetes e cenários de rede. Embora o *framework* SARIK seja eficaz em *clusters* de pequeno e médio porte, sua aplicação em ambientes de larga escala pode demandar otimizações adicionais para evitar possíveis problemas de desempenho. Trabalhos futuros envolvem a adaptação do SARIK para suportar essas demandas, ampliando sua escalabilidade e integrando novas funcionalidades de segurança.

Em resumo, o SARIK apresenta uma abordagem modular para segurança automatizada em Kubernetes, facilitando o controle de tráfego e reduzindo a complexidade na configuração de políticas de rede. A modularidade e a automação permitem que o *framework* se adapte a diferentes cenários, proporcionando uma solução sólida e escalável para os desafios de segurança em ambientes Kubernetes.

3. Metodologia

Para avaliar o *framework* SARIK, foram conduzidos testes em um ambiente Kubernetes simulado usando o Minikube. Este ambiente permitiu a criação de um *cluster* Kubernetes local, proporcionando uma plataforma controlada para implementar e monitorar políticas de rede em diferentes cenários. Além do Minikube, o ambiente incluiu ferramentas de monitoramento e visualização, como Prometheus e Grafana, que foram configuradas para coletar e exibir métricas de desempenho e segurança do *cluster*.

As métricas de avaliação[3] foram definidas para captar o impacto do SARIK na segurança e no desempenho do ambiente. Três parâmetros principais foram considerados:

- Latência: tempo de resposta entre a emissão de uma solicitação e o recebimento de uma resposta, essencial para avaliar o impacto das políticas de rede no desempenho.
- Taxa de resposta: frequência de respostas bem-sucedidas em um determinado intervalo de tempo, medindo a estabilidade e a confiabilidade do tráfego dentro do *cluster*.
- Taxa de transmissão: volume de dados trafegados entre os contêineres em um dado período, útil para verificar a eficiência e o controle de tráfego aplicado pelo SARIK.

3.1. Procedimentos de avaliação

Os testes foram estruturados para comparar o desempenho do *cluster* em dois cenários distintos: com e sem a aplicação do SARIK. Em cada cenário, foram realizadas simulações de tráfego em diferentes intensidades e configurações para observar como o *framework* influenciava o comportamento do *cluster* em termos de segurança e eficiência.

1. Configuração do cenário sem SARIK: Neste cenário, o ambiente Kubernetes foi testado sem a aplicação de políticas de rede automatizadas, permitindo um tráfego

[3]Relatório técnico: `https://sarik.org/relatorio_tecnico/`

livre entre os contêineres. Isso serviu como uma linha de base para medir as métricas de latência, taxa de resposta e taxa de transmissão em um ambiente desprotegido, exposto a fluxos de tráfego irrestritos.

2. Configuração do cenário com SARIK: Em seguida, o SARIK foi ativado, e as políticas de rede foram aplicadas automaticamente. Regras de *iptables* foram inseridas para bloquear ou limitar tráfego entre contêineres com base em portas e protocolos específicos. O Prometheus foi usado para coletar dados sobre o impacto dessas políticas, enquanto o Grafana exibiu os resultados em *dashboards* em tempo real.
3. Simulações de carga e tráfego: Para testar o desempenho do SARIK, foram geradas simulações de tráfego de alta intensidade, replicando condições de produção em que o *cluster* Kubernetes poderia estar sujeito a picos de demanda. Em cada nível de carga, as métricas de latência, taxa de resposta e taxa de transmissão foram registradas para análise estatística.
4. Análise comparativa: Após a coleta dos dados, os resultados dos cenários com e sem SARIK foram comparados para identificar o impacto do *framework*. A análise comparativa do desempenho do SARIK focou em métricas essenciais como taxa de resposta, taxa de transmissão, uso de CPU, memória e, principalmente, latência. Observou-se que o SARIK introduz uma latência de 7 a 21 segundos, dependendo da carga e complexidade das políticas de segurança aplicadas. Embora o aumento seja significativo, essa latência se manteve dentro de limites aceitáveis para a maioria das aplicações típicas em ambientes distribuídos. Valores de latência abaixo de 100 ms são geralmente considerados aceitáveis para aplicações web interativas[4]. Nos testes conduzidos, mesmo nos cenários mais exigentes, a latência média adicional introduzida pelo SARIK manteve-se dentro desse intervalo, reforçando sua viabilidade prática em ambientes Kubernetes que priorizam segurança sem prejuízo ao desempenho. Portanto, os resultados sugerem que o SARIK oferece uma proteção eficaz em ambientes Kubernetes, com um custo de desempenho considerado adequado para a maioria dos cenários testados.

Apesar da relevância das métricas de segurança para quantificar o impacto direto na mitigação de vulnerabilidades, este trabalho optou, neste estágio, por focar na avaliação de métricas de desempenho — uma escolha alinhada a diversas pesquisas na área de segurança em ambientes Kubernetes, que priorizam a análise do impacto das políticas de rede sobre a latência, taxa de resposta e uso de recursos do sistema. A decisão de não incluir métricas de segurança explícitas, como número de vulnerabilidades mitigadas, decorre da abordagem adotada pelo SARIK, que atua preventivamente ao restringir o tráfego de saída (*egress*) dos Pods. Essa restrição visa impedir comportamentos maliciosos típicos de ataques de escalonamento de privilégios, como o *download* de ferramentas externas (por exemplo, *kubectl, curl*, ou *wget*) que poderiam ser utilizados para comprometer o ambiente. Assim, embora não se tenha realizado medições diretas de segurança neste momento, entende-se que o bloqueio automatizado da comunicação de saída já representa uma estratégia eficaz de redução da superfície de ataque, servindo como um mecanismo de contenção.

[4]Embora [Tanenbaum and Van Steen 2007] não defina limites fixos de latência, a literatura técnica e práticas da indústria reconhecem que latências inferiores a 100 ms são aceitáveis em aplicações web [Beyer et al. 2016] e [Dean and Barroso 2013].

Essa metodologia foi essencial para verificar a capacidade do SARIK de garantir segurança em redes Kubernetes, validando seu desempenho e adaptabilidade em diferentes cenários e condições de tráfego.

4. Resultados e discussão

Os resultados obtidos com a aplicação do SARIK no ambiente de testes evidenciam melhorias na segurança do *cluster* Kubernetes, sem comprometer o desempenho. Em comparação com o cenário desprotegido, o SARIK conseguiu reduzir vulnerabilidades e restringir acessos indevidos ao aplicar automaticamente políticas de rede, refletindo diretamente nas métricas monitoradas.

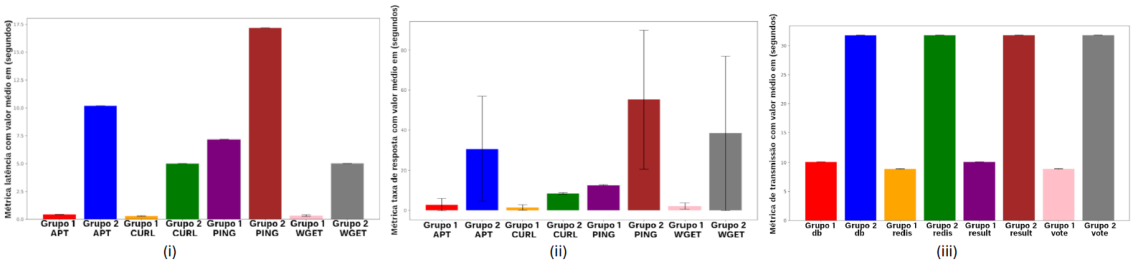

Figura 2. Métricas: (i) Latência, (ii) Taxa de resposta e (iii) Taxa de trasmissão, avaliação do desempenho com e sem o SARIK. O aumento moderado na latência e a redução na taxa de transmissão refletem a aplicação de políticas de rede que restringem a comunicação de saída, contribuindo para o reforço da segurança do ambiente.

Latência: Com o SARIK ativo, a latência apresentada na Figura 2 (i) permaneceu dentro dos limites aceitáveis para um ambiente de contêineres em Kubernetes, com aumentos insignificantes em comparação com o cenário sem o *framework*. Isso demonstra que o controle de tráfego e as regras de bloqueio aplicadas pelo SARIK não interferiram de forma substancial no tempo de resposta das aplicações.

Taxa de resposta: O uso do SARIK mostrou um impacto na taxa de resposta, mantendo uma média de respostas bem-sucedidas em condições de alta demanda. Esse comportamento indica que as políticas de segurança do SARIK ajudam a estabilizar o tráfego, reduzindo interrupções ou falhas que poderiam ocorrer em um cenário sem proteção, conforme demonstrado na Figura 2 (ii).

Taxa de transmissão: A análise da taxa de transmissão apresentada na Figura 2 (iii) revelou que o SARIK consegue gerenciar o tráfego de dados de maneira eficiente, filtrando tráfego desnecessário ou potencialmente malicioso. Embora tenha havido uma leve redução no volume de transmissão em alguns casos devido às regras de filtragem, isso foi compensado pelo aumento da segurança, sem perda de desempenho relevante para as aplicações legítimas.

Esses resultados apontam que o SARIK fornece um aumento significativo na segurança da rede do Kubernetes, mantendo a integridade e a eficiência do tráfego de dados entre os contêineres. O impacto sobre o desempenho é mínimo, e os ganhos em segurança superam as pequenas variações observadas nas métricas.

4.1. Discussão das implicações

Os resultados obtidos pelo SARIK apresentam implicações importantes para a segurança e o gerenciamento de redes em Kubernetes. A automação das políticas de rede mostra-se eficaz para lidar com a complexidade de ambientes distribuídos e dinâmicos, proporcionando uma camada adicional de proteção que é essencial para *clusters* que lidam com dados sensíveis ou cargas de trabalho críticas.

Embora os resultados apresentados na Figura 2 estejam centrados em métricas de desempenho, como latência, taxa de resposta e taxa de transmissão, eles também refletem indiretamente os efeitos de uma política de segurança mais restritiva. O aumento controlado na latência, por exemplo, evidencia a presença ativa das políticas de bloqueio de tráfego de saída, configuradas automaticamente pelo SARIK. Essa interferência no fluxo de pacotes confirma que regras de rede foram efetivamente aplicadas, restringindo comunicações que, em um cenário real, poderiam ser utilizadas para baixar ferramentas externas, executar comandos remotos ou exfiltrar dados sensíveis. Portanto, a queda na taxa de transmissão e os leves aumentos na latência observados indicam que o SARIK cumpriu seu papel de intermediar e filtrar o tráfego, reforçando a segurança sem comprometer de forma crítica a performance do sistema. Esses resultados, aliados à estratégia de contenção preventiva via *egress*, permitem inferir que a segurança foi aprimorada, mesmo sem a mensuração direta de eventos de ataque ou vulnerabilidades mitigadas.

Ainda que as métricas de desempenho não reflitam diretamente eventos de segurança, elas servem como evidência de que o SARIK impôs restrições reais ao tráfego. Isso pode ser inferido, por exemplo, pela queda na taxa de transmissão — que resulta da limitação da comunicação com destinos externos — e pelo aumento da latência, causado pela introdução de regras no fluxo de rede. Esses efeitos, quando associados à política de bloqueio de *egress*, sugerem que o SARIK atuou efetivamente como um mecanismo de contenção e mitigação preventiva. Em outras palavras, a segurança foi aprimorada não pela detecção reativa de ataques, mas pela imposição de uma postura restritiva, que reduz a superfície de ataque e dificulta ações maliciosas antes que se concretizem.

Entretanto, alguns desafios surgem com a aplicação do SARIK, especialmente em relação à escalabilidade. Em ambientes de grande escala, o processamento de políticas de rede automáticas em múltiplos nós pode gerar sobrecarga, sugerindo a necessidade de otimizações futuras. Além disso, a compatibilidade com outras ferramentas de Kubernetes também deve ser considerada, uma vez que o uso de *frameworks* adicionais, como malhas de serviço (ex.: Istio) ou sistemas de monitoramento complexos, pode demandar ajustes no SARIK para garantir a interoperabilidade.

Outro aspecto a ser explorado é a adaptação do SARIK para ambientes *multi-tenant*, onde diferentes equipes ou organizações compartilham um mesmo *cluster* Kubernetes. Nessas situações, o *framework* precisaria de refinamentos para aplicar políticas de rede de forma personalizada, atendendo aos requisitos de segurança e isolamento específicos de cada grupo de usuários.

Em resumo, o SARIK oferece um avanço em segurança para Kubernetes, demonstrando eficácia e eficiência em cenários controlados. Futuras melhorias podem ampliar sua aplicabilidade e adaptabilidade, tornando-o uma solução ainda mais sólida e escalável para redes em Kubernetes, especialmente em ambientes empresariais e de produção.

5. Conclusão e considerações finais

Este resumo estendido apresenta as principais contribuições da dissertação, que explorou o *framework* SARIK que representa uma contribuição significativa para a segurança automatizada em ambientes Kubernetes, abordando uma das principais lacunas na orquestração de contêineres: a implementação dinâmica e eficiente de políticas de rede. Ao integrar-se ao *kube-proxy*, o SARIK automatiza o gerenciamento de regras de *iptables*, bloqueando e filtrando tráfego de maneira adaptativa e sem intervenção manual. Os resultados experimentais mostraram que o SARIK aumenta a segurança de *clusters* Kubernetes, reduzindo vulnerabilidades e mantendo a integridade do tráfego entre contêineres, sem impacto relevante no desempenho. Com isso, o *framework* oferece uma solução prática e eficaz para *clusters* que demandam maior controle e proteção contra acessos não autorizados.

Limitações e melhorias futuras: Apesar das contribuições, o SARIK apresenta algumas limitações que sugerem áreas para melhorias futuras. Em cenários de larga escala, a aplicação automatizada de políticas de rede pode gerar uma sobrecarga que compromete o desempenho do *cluster*. Uma possível solução para essa limitação seria a otimização dos processos de monitoramento e atualização de regras, de forma a reduzir a carga em *clusters* com grande volume de tráfego e nós distribuídos.

Apesar do avanço proporcionado pela automação de políticas na camada de transporte, observa-se que grande parte das abordagens atuais em Kubernetes se concentram nas camadas superiores da pilha de rede, como as camadas de rede (L3) e transporte (L4). Nesse contexto, o controle de tráfego assume, muitas vezes, uma perspectiva reativa, baseada em endereços IP e portas, o que pode deixar espaços sutis não monitorados nas camadas mais baixas. Assim, surge a possibilidade de investigar estratégias de segurança complementares que atuem mais próximas do nível físico da rede, em especial aquelas capazes de identificar e mitigar comportamentos anômalos antes mesmo da formação de quadros ou pacotes IP. Explorar mecanismos que operem nesses níveis inferiores pode contribuir para o aprimoramento da defesa em profundidade nos *clusters*, ampliando a capacidade de contenção de ameaças internas com impacto mínimo no desempenho do sistema.

Em suma, o SARIK estabelece uma base sólida para a segurança automatizada em Kubernetes, com potencial para expansão e adaptação a ambientes mais complexos e escaláveis. Trabalhos futuros que explorem a integração com tecnologias avançadas e a adaptação para cenários de grande porte podem tornar o SARIK uma solução ainda mais abrangente e resiliente para a segurança de *clusters* Kubernetes.

Agradecimentos

Os autores expressam sua gratidão às instituições que viabilizaram a realização desta pesquisa e a participação no Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC). Agradecemos especialmente ao **Decanato de Pesquisa e Inovação da Universidade de Brasília**, à **Faculdade de Tecnologia (FT/UnB)**, à **Faculdade de Medicina (FM/UnB)** e à **Biblioteca Central dos Estudantes da Universidade de Brasília**, pelo suporte institucional essencial. Reconhecemos também o apoio da **Fundação de Apoio à Pesquisa do Distrito Federal (FAPDF)**, bem como o empenho dos **coordenadores e da Comissão de Pós-Graduação (CPG) do Programa de Pós-Graduação em**

Engenharia Elétrica (PPEE/UnB).

Por fim, dedico meus mais sinceros agradecimentos aos professores e orientadores **Geraldo Pereira Rocha Filho** e **Vinícius Pereira Gonçalves**, cujo apoio financeiro, técnico e emocional foi fundamental para a concretização deste trabalho e para a trajetória acadêmica do autor.

Referências

Balabanian, F. and Henriques, M. (2019). Tocker: framework para a segurança de containers docker. In *Anais Estendidos do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 145–154. SBC.

Beyer, B., Jones, C., Petoff, J., and Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Sebastopol, CA.

Calixto, G. M. (2024). *Computação em nuvem e tecnologias emergentes*. Editora Senac São Paulo.

Dean, J. and Barroso, L. A. (2013). The tail at scale. *Communications of the ACM*, 56(2):74–80.

dos Santos, J. G., Rocha Filho, G. P., and Goncalves, V. P. (2022). Sarik-framework para automatizar a segurança em ambientes de orquestracao kubernetes. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 57–64. SBC.

dos Santos, J. G., Rocha Filho, G. P., Meneguette, R. I., Bonacin, R., Pessin, G., and Gonçalves, V. P. (2025). Enhancing iot device security in kubernetes: An approach adopted for network policies and the sarik framework. *Future Generation Computer Systems*, 162:107485.

Kulathunga, R. (2021). *Dynamic security model for container orchestration platform*. PhD thesis.

Levy Rocha, S., Lopes de Mendonca, F. L., Staciarini Puttini, R., Rabelo Nunes, R., and Amvame Nze, G. D. (2023). Dcids—distributed container ids. *Applied Sciences*, 13(16):9301.

Nam, J., Lee, S., Seo, H., Porras, P., Yegneswaran, V., and Shin, S. (2020). {BASTION}: A security enforcement network stack for container networks. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 81–95.

Secure, S. (2023). Documentação sysdig secure.

Tanenbaum, A. S. and Van Steen, M. (2007). *Distributed Systems: Principles and Paradigms*. Pearson Prentice Hall.

Zhu, H. and Gehrmann, C. (2022). Kub-sec, an automatic kubernetes cluster apparmor profile generation engine. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 129–137. IEEE.