FACULDADE DE TECNOLOGIA DE AMERICANA "Ministro Ralph Biasi"
Curso Superior de Tecnologia em Segurança da Informação

Claudio Kenji Terassaka

David Henrique da Silva Carnaval

TRANSMISSÃO SEGURA DE DADOS PESSOAIS E SENSÍVEIS EM SISTEMAS DE COMUNICAÇÃO ASSÍNCRONA

Americana, SP

2026

FACULDADE DE TECNOLOGIA DE AMERICANA "Ministro Ralph Biasi"
Curso Superior de Tecnologia em Segurança da Informação

Claudio Kenji Terassaka

David Henrique da Silva Carnaval

TRANSMISSÃO SEGURA DE DADOS PESSOAIS E SENSÍVEIS EM SISTEMAS DE COMUNICAÇÃO ASSÍNCRONA

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação sob a orientação do Prof. Marcus Vinicius Lahr Giraldi.

Área de concentração: Segurança da Informação.

Americana, SP

2026

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana Ministro Ralph Biasi- CEETEPS Dados Internacionais de Catalogação-na-fonte

TERASSAKA, Claudio Kenji

Transmissão segura de dados pessoais e sensíveis em sistemas de comunicação assíncrona. / Claudio Kenji Terassaka, David Henrique da Silva Carnaval – Americana, 2026.

49f.

Monografia (Curso Superior de Tecnologia em Segurança da Informação) - - Faculdade de Tecnologia de Americana Ministro Ralph Biasi – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Esp. Marcus Vinícius Lahr Giraldi

1. LGPD – Lei Geral de Proteção de Dados 2. Segurança em sistemas de informação. I. TERASSAKA, Claudio Kenji, II. CARNAVAL, David Henrique da Silva III. GIRALDI, Marcus Vinícius Lahr IV. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana Ministro Ralph Biasi

CDU: 34:681.3
681.518.5

Elaborada pelo autor por meio de sistema automático gerador de ficha catalográfica da Fatec de Americana Ministro Ralph Biasi.

FOLHA DE APROVAÇÃO

Claudio Kenji Terassaka
David Henrique da Silva Carnaval
[redacted]

TRANSMISSÃO SEGURA DE DADOS PESSOAIS E SENSÍVEIS EM SISTEMAS DE COMUNICAÇÃO ASSÍNCRONA

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo Centro Paula Souza – Faculdade de Tecnologia de Americana – Ministro Ralph Biasi.
Área de concentração: Segurança da Informação.

Americana, 09 de junho de 2026

Banca Examinadora:

Marcus Vinicius Lahr Giraldi
Especialista
Faculdade de Tecnologia de Americana

Lídia Regina de Carvalho Freitas Barban
Especialista
Faculdade de Tecnologia de Americana

Henri Alves de Godoy
Doutor
Faculdade de Tecnologia de Americana

RESUMO

O presente trabalho tem como foco analisar e definir uma abordagem segura para transmissão de dados pessoais e sensíveis em sistemas de comunicação assíncronas, com foco na proteção dos dados transmitidos em ambientes empresariais. Amplamente adotada em sistemas distribuídos, como plataformas de pagamento, serviços financeiros e *streaming*, a comunicação assíncrona oferece escalabilidade e flexibilidade, mas enfrenta desafios significativos diante de ameaças cibernéticas, como interceptação de mensagens e ataques man-in-the-middle, entre outros. Este estudo busca identificar mecanismos eficazes para garantir a confidencialidade e a integridade das informações trocadas, considerando o aumento da dependência de comunicações confiáveis em todos os ramos da sociedade. A metodologia consiste em uma pesquisa exploratória e bibliográfica, baseada na revisão de artigos acadêmicos, *whitepapers* e documentação técnica de ferramentas baseados na comunicação assíncrona como Kafka e RabbitMQ. A análise aborda técnicas como criptografia de ponta a ponta, assinaturas digitais e políticas de entrega (*at-least-once* e *exactly-once*), visando mitigar os riscos em cenários de operação normalizados, alta latência ou instabilidade de rede. Espera-se com este trabalho mapear boas práticas de segurança aplicáveis a sistemas críticos e avaliar a eficácia dessas soluções na proteção contra vulnerabilidades comuns, além de distribuir um padrão encapsulado e pronto para ser usado nos diversos ambientes. A relevância deste trabalho reside na crescente necessidade de sistemas de *mensageria* seguros, que assegurem a privacidade dos dados e a confiança de usuários e empresas em um contexto de evolução tecnológica e ameaças digitais.

Palavras-chave: Dados sensíveis e pessoais**;** Comunicação assíncrona e síncrona; *Mensageria*, Ataques.

ABSTRACT

This paper focuses on analyzing and defining a secure approach for transmitting personal and sensitive data in asynchronous communication systems, with a focus on protecting transmitted data in enterprise environments. Widely adopted in distributed systems such as payment platforms, financial services, and streaming, asynchronous communication offers scalability and flexibility, but faces significant challenges in the face of cyber threats, such as message interception and man-in-the-middle attacks. This study seeks to identify effective mechanisms to ensure the confidentiality and integrity of exchanged information, considering the increasing reliance on reliable communications in all sectors of society. The methodology consists of exploratory and bibliographic research, based on a review of academic articles, whitepapers, and technical documentation for tools such as Kafka and RabbitMQ. The analysis addresses techniques such as end-to-end encryption, digital signatures, and delivery policies (at-least-once and exactly-once), aiming to mitigate risks in scenarios involving normalized operation, high latency, or network instability. This work aims to map best security practices applicable to critical systems and evaluate the effectiveness of these solutions in protecting against common vulnerabilities. It also aims to distribute an encapsulated standard ready for use in various environments. The relevance of this work lies in the growing need for secure messaging systems that ensure data privacy and the trust of users and businesses in a context of technological evolution and digital threats.

Keywords: *Personal and Sensitive data; Synchronous and Asynchronous communication; Messaging.*

LISTA DE FIGURAS

Figura 1 - Comunicação segura no Apache Kafka utilizando TLS/SSL e certificados digitais para criptografia dos dados em trânsito. 29
Figura 2 - Configuração RabbitMQ da porta 5671. 31
Figura 3 - Configuração do RabbitMQ com a porta 5672 insegura ativa. 32
Figura 4 - Isolamento de containers e restrição de acesso externo em rede interna Docker. 34
Figura 5 - Monitoramento de eventos de autenticação e detecção de anomalias utilizando Elastic Security. 35
Figura 6 - Configuração de ambiente RabbitMQ a partir do Docker Desktop. ... 36
Figura 7 - Captura de tráfego no Wireshark demonstrando transmissão insegura de dados pelo RabbitMQ. 37
Figura 8 - Captura do terminal do RabbitMQ demonstrando nome capturado no Wireshark. 37
Figura 9 - Configuração de ambiente Kafka a partir do Docker Desktop. 38
Figura 10 - Captura de tráfego no Wireshark demonstrando transmissão insegura de dados pelo Kafka. 38
Figura 11 - Captura de tráfego TLS 1.2 no Wireshark demonstrando handshake criptográfico e transmissão segura de dados. 39

SUMÁRIO

1. INTRODUÇÃO 7
2. FUNDAMENTAÇÃO TEÓRICA 9
2.1. Comunicação síncrona e assíncrona 9
2.2. Sistemas de Mensageria 11
2.3. Padrões de entrega de mensagens 12
2.4. Dados pessoais e sensíveis 13
3. METODOLOGIA 16
3.1. Etapas da pesquisa 17
4. DESENVOLVIMENTO 19
4.1. Problemática 19
4.2. Tecnologias Envolvidas 22
4.3. Segurança em Mensageria 23
4.3.1. Criptografia em trânsito (TLS/SSL) 24
4.3.2. Criptografia em repouso 24
4.3.3. Autenticação 25
4.3.4. Autorização (Controle de acesso) 26
4.3.5. Integridade das mensagens 26
4.3.6. Principais vulnerabilidades em mensageria 27
4.4. Prática 28
4.4.1. Ambiente de Testes 28
4.4.2. Configuração do Apache Kafka 28
4.4.3. Configuração do RabbitMQ 30
4.4.4. Configuração de Rede e Isolamento 33
4.4.5. Logs e Monitoramento 34
4.4.6. Testes Realizados 35
4.4.7. Disponibilização do Ambiente 39
4.5. Ganhos de Segurança 39
5. CONCLUSÃO 43
6. REFERÊNCIAS 45

1. INTRODUÇÃO

A comunicação assíncrona, amplamente empregada em sistemas distribuídos, exemplo de e-mails, filas de mensagens e aplicações empresariais, distingue-se pela independência temporal entre dois sistemas que precisam se comunicar. Essa característica possibilita que o envio e o recebimento ocorram em momentos distintos através do agrupamento de mensagens em uma fila, conferindo ao sistema uma menor dependência a outro. Tal abordagem oferece vantagens relevantes, sobretudo no que diz respeito à escalabilidade, à resiliência a falhas e à adaptabilidade, sendo particularmente valiosa em arquiteturas baseadas em microsserviços, ambientes colaborativos e sistemas corporativos de grande escala (KLEPPMANN, 2017).

No entanto, à medida que cresce a adoção desses sistemas, especialmente em domínios que lidam com dados sensíveis, como informações pessoais, financeiras, de saúde ou governamentais, torna-se imperativa a adoção de mecanismos robustos de segurança na transmissão destas mensagens. A ausência de tais medidas expõe as comunicações a riscos como interceptações, vazamentos, adulterações e acessos não autorizados, comprometendo princípios fundamentais da segurança da informação, como a confidencialidade, a integridade e a autenticidade das informações (OWASP FOUNDATION, 2021).

Nesse contexto, este trabalho tem por finalidade explorar os principais mecanismos de proteção voltados à segurança de dados sensíveis transmitidos por meio de sistemas de *mensageria* assíncrona. A pesquisa contempla o estudo de técnicas como criptografia de ponta a ponta, autenticação de mensagens, uso de certificados digitais, controles de acesso e práticas recomendadas de configuração segura em plataformas amplamente utilizadas, como RabbitMQ e Apache Kafka.

A importância desta investigação reside na crescente necessidade por soluções de *mensageria* que sejam, simultaneamente, seguras e confiáveis, exigência que se intensifica frente a legislações como a Lei Geral de Proteção de Dados (LGPD) e ao avanço contínuo das ameaças cibernéticas. Busca-se, assim, oferecer uma análise crítica das práticas de segurança atualmente adotadas, evidenciando suas fragilidades e sugerindo estratégias para o aprimoramento da proteção de dados em trânsito (BRASIL, 2018) .

A viabilidade da pesquisa está assegurada pela existência de vasta literatura especializada, ferramentas consolidadas no mercado e estudos acadêmicos que tratam do tema. Com base nesse alicerce, foram definidos os objetivos gerais e específicos que nortearão o desenvolvimento da investigação.

Além dos riscos teóricos, a conjuntura atual revela um aumento expressivo em incidentes ligados à exposição de dados em movimento e a falhas em sistemas distribuídos. Estudos de segurança cibernética lançados entre 2024 e 2025 indicam uma ascensão em ataques do tipo Man-in-the-Middle (MITM), vazamentos decorrentes de configurações incorretas em intermediários de mensagens e a exposição indevida de ambientes Kafka e RabbitMQ abertos ao público. Em várias situações, a falta de criptografia TLS, autenticação robusta ou isolamento de redes facilitou o acesso não autorizado a comunicações com dados pessoais, financeiros e credenciais de sistemas empresariais. Tais eventos sublinham que a proteção em sistemas de comunicação assíncrona precisa ser considerada um pilar fundamental, e não um complemento, para assegurar a aderência a normas, a salvaguarda da privacidade e a operação contínua das empresas (TANENBAUM; VAN STEEN, 2019; KLEPPMANN, 2017).

2. FUNDAMENTAÇÃO TEÓRICA

A seguir, serão apresentados os principais conceitos relacionados à comunicação assíncrona e à segurança em sistemas de *mensageria* utilizados em arquiteturas distribuídas. Inicialmente, serão abordadas as diferenças entre comunicação síncrona e assíncrona, destacando suas características, vantagens e aplicações em ambientes modernos de software. Em seguida, serão apresentados os sistemas de *mensageria*, com foco nas plataformas Apache Kafka e RabbitMQ, amplamente utilizadas para troca de mensagens e processamento de eventos em tempo real. Posteriormente, serão discutidos os conceitos relacionados aos padrões de entrega de mensagens e à transmissão de dados pessoais e sensíveis, enfatizando aspectos ligados à segurança da informação, confidencialidade e conformidade com legislações como a LGPD. Por fim, serão apresentados os principais mecanismos de proteção aplicados em ambientes distribuídos, incluindo criptografia, autenticação, autorização e controle de acesso.

2.1. Comunicação síncrona e assíncrona

A comunicação pode ocorrer de forma síncrona ou assíncrona, sendo cada uma caracterizada por diferentes formas de interação entre os participantes. A comunicação síncrona é marcada pela troca de informações em tempo real, na qual emissor e receptor interagem simultaneamente. Esse tipo de comunicação ocorre, por exemplo, em reuniões presenciais, chamadas de vídeo, telefonemas ou chats ao vivo. Sua principal vantagem está na imediaticidade das respostas, possibilitando que dúvidas sejam esclarecidas e decisões tomadas de forma mais ágil. Segundo Tanenbaum (2019, p. 7), no livro Sistemas Distribuídos, “nessa forma de comunicação, uma parte que requisita um serviço, em geral denominada cliente, fica bloqueada até que uma mensagem seja enviada de volta”. Ou seja, no contexto tecnológico, a comunicação síncrona exige que o processo solicitante aguarde a resposta para dar continuidade, o que reforça a ideia de dependência temporal entre as partes envolvidas (TANENBAUM; VAN STEEN, 2019; ZENDESK, 2024).

Por outro lado, a comunicação assíncrona caracteriza-se pela ausência de dependência temporal. No contexto de sistemas, significa que um cliente pode enviar uma solicitação a um servidor sem precisar esperar ativamente pela resposta, ficando livre para executar outras tarefas em paralelo. O servidor, por sua vez, processa o pedido assim que possível e, posteriormente, envia a resposta por meio de um canal adequado. Isso só é possível graças a padrões de *software* baseados em eventos (*event-driven*), que estruturam a comunicação por meio da produção (*publisher*), detecção e consumo de eventos (*subscriber*), geralmente mediados por um intermediário. Cada evento pode representar tanto uma notificação de que algo ocorreu quanto uma solicitação a ser processada de forma resiliente (KREPS; NARKHEDE; RAO, 2011; RESULTADOS DIGITAIS, 2024).

Além da aplicação em sistemas computacionais, a comunicação assíncrona também se manifesta em práticas sociais e culturais. Historicamente, ela se evidencia nas pinturas rupestres, nas cartas escritas à mão e em livros, em que a interação não ocorre diretamente entre autor e leitor, mas por meio da mensagem registrada. No cenário contemporâneo, encontra-se em *e-mails*, fóruns, redes sociais e caixas postais. Em todos esses exemplos, a principal característica é o desacoplamento temporal: não há uma interação direta entre quem cria a mensagem e quem a consome, mas sim a interação do receptor com o conteúdo transmitido.

A escolha entre comunicação síncrona e assíncrona depende do contexto e das necessidades específicas. A comunicação síncrona é vantajosa quando se exige resposta imediata, interação direta e decisões rápidas, mas apresenta como desvantagens a necessidade de disponibilidade simultânea e o risco de sobrecarga em ambientes descentralizados. Já a comunicação assíncrona oferece maior flexibilidade, escalabilidade e melhor aproveitamento de recursos, mas pode resultar em atrasos na resposta e maior complexidade no controle do fluxo de informações. Em sistemas distribuídos, ambos os modelos são complementares: a comunicação síncrona tende a ser utilizada em operações críticas e de curta duração, enquanto a assíncrona é preferida em processos que demandam resiliência, desacoplamento e alta escalabilidade (KREPS; NARKHEDE; RAO, 2011; RESULTADOS DIGITAIS, 2024).

2.2. Sistemas de *Mensageria*

Os sistemas de *mensageria* são ferramentas que viabilizam a comunicação entre aplicações, serviços ou usuários, permitindo a troca de informações de forma síncrona ou, mais comumente, assíncrona. Eles desempenham papel central em arquiteturas distribuídas ao oferecer um meio confiável de transmissão de dados, desacoplando produtores e consumidores de mensagens. O funcionamento segue um fluxo no qual um produtor envia um evento para um *broker*, que atua como intermediário entre cliente e servidor. Esse evento, transformado em mensagem, é armazenado sequencialmente em uma fila até ser consumido pelo destinatário, garantindo maior flexibilidade e resiliência nos sistemas.

Entre os exemplos mais relevantes de plataformas de *mensageria* destacam-se RabbitMQ e ActiveMQ, amplamente utilizados em integrações corporativas por seu suporte a protocolos como AMQP e pela robustez no gerenciamento de filas. Outra ferramenta, o Apache Kafka, que de acordo com a Apache Software Foundation (2026), Kafka "*is an open-source distributed event streaming platform*". o Kafka é uma solução *open source* tolerante a falhas, projetada para lidar com grandes quantidades de mensagens em tempo reduzido, sendo altamente indicado para aplicações em microsserviços e cenários que demandam comunicação em tempo real (APACHE SOFTWARE FOUNDATION, 2026; VMWARE, 2026).

Esses sistemas possibilitam diferentes padrões de comunicação, entre os quais se destacam o modelo *point-to-point* (P2P), no qual uma mensagem é consumida por apenas um destinatário, e o modelo *publish-subscribe* (*pub-sub*), em que uma mensagem publicada em um tópico pode ser recebida por múltiplos assinantes. Essa flexibilidade amplia o alcance das aplicações de *mensageria*, tornando-as essenciais em áreas como serviços financeiros, comércio eletrônico e *Internet* das Coisas, onde a escalabilidade e a confiabilidade na transmissão de dados são fundamentais.

Apesar das vantagens, a adoção de sistemas de *mensageria* também apresenta desafios, como a complexidade de configuração e manutenção, a necessidade de estratégias robustas para garantir alta disponibilidade e a preocupação com a segurança das mensagens em trânsito. Ainda assim, quando bem planejados, esses sistemas se consolidam como componentes indispensáveis para

arquiteturas modernas, fornecendo a base necessária para integração eficiente de sistemas e o processamento de dados em grande escala.

2.3. Padrões de entrega de mensagens

Nos sistemas de comunicação assíncrona, especialmente em arquiteturas orientadas a eventos e *mensageria*, a forma como as mensagens são entregues é fundamental para garantir a confiabilidade do processo. Nesse contexto, surgem três padrões principais: *At most once*, *At least once* e *Exactly once*, cada um com características próprias e diferentes implicações em termos de desempenho, complexidade e tolerância a falhas.

O padrão *At most once* garante que a mensagem seja enviada no máximo uma vez, sem novas tentativas em caso de falha. Isso significa que não haverá duplicações, mas, por outro lado, pode haver perda de mensagens. É indicado para cenários nos quais a perda de alguns dados não comprometa o funcionamento do sistema, como notificações não críticas ou métricas de monitoramento.

Já o padrão *At least once* assegura que a mensagem será entregue ao menos uma vez, ainda que, em situações de falha, o destinatário possa recebê-la mais de uma vez. Essa abordagem prioriza a confiabilidade em detrimento da duplicação, sendo adequada para sistemas em que a perda de mensagens é inaceitável, como em processos de faturamento, registros de transações ou comunicação entre serviços financeiros. Para lidar com duplicações, é comum implementar mecanismos de independência, garantindo que uma mesma operação processada mais de uma vez não gera inconsistências.

Por fim, o padrão *Exactly once* busca garantir que cada mensagem seja entregue uma única vez ao destinatário, sem perdas nem duplicações. Trata-se do modelo mais desejado, mas também o mais complexo e custoso de implementar, pois exige controles adicionais de estado e coordenação entre remetente e receptor. É adotado em sistemas críticos que não toleram erros de entrega nem duplicações, como transferências bancárias ou atualizações de dados sensíveis em cadastros de clientes.

Dessa forma, a escolha entre os padrões deve considerar o equilíbrio entre confiabilidade, desempenho e complexidade, sendo fundamental analisar o contexto de aplicação. Em sistemas que tratam dados pessoais e sensíveis, por exemplo, a decisão impacta diretamente na segurança, na consistência das informações e na experiência do usuário (CORRÊA, 2022; KLEPPMANN, 2017).

2.4. Dados pessoais e sensíveis

No contexto da sociedade da informação, os dados tornaram-se um dos principais ativos econômicos e estratégicos. Contudo, nem todos os dados possuem o mesmo grau de impacto ou de risco em caso de mau uso. Por essa razão, legislações nacionais e internacionais passaram a distinguir diferentes categorias de dados, entre elas os dados pessoais e os dados pessoais sensíveis, estabelecendo níveis distintos de proteção e exigências para seu tratamento.

De acordo com a Lei nº 13.709/2018, conhecida como LGPD, considera-se dado pessoal “informação relacionada a pessoa natural identificada ou identificável” (BRASIL, 2018). Essa definição é ampla e abrange qualquer informação que permita a identificação direta (como nome, CPF, RG) ou indireta (como endereço, e-mail ou localização geográfica) de um indivíduo. Assim, mesmo dados que isoladamente não identificam uma pessoa podem ser considerados pessoais se, combinados com outros, possibilitarem a identificação (BRASIL, 2018).

Ainda segundo a LGPD, dado pessoal sensível é aquele que envolve informações cuja utilização indevida pode gerar riscos mais elevados à liberdade individual, à privacidade ou até à integridade da pessoa. A lei define como sensíveis os “dados sobre origem racial ou étnica, convicção religiosa, opinião política, filiação a sindicato ou a organização de caráter religioso, filosófico ou político, dado referente à saúde ou à vida sexual, dado genético ou biométrico, quando vinculado a uma pessoa natural” (BRASIL, 2018). Essa categoria demanda salvaguardas mais rigorosas, uma vez que sua exposição pode resultar em discriminação, estigmatização ou violações graves de direitos fundamentais.

A distinção entre essas categorias também é encontrada no Regulamento Geral de Proteção de Dados da União Europeia (GDPR – *General Data Protection*

Regulation, 2016). O GDPR define dados pessoais como qualquer informação relacionada a uma pessoa identificada ou identificável, abrangendo inclusive identificadores digitais, como endereços de IP e *cookies*. Quanto aos dados sensíveis, denominados “categorias especiais de dados”, incluem origem racial ou étnica, opiniões políticas, convicções religiosas ou filosóficas, filiação sindical, dados genéticos, dados biométricos (para identificação), dados relativos à saúde e à vida sexual ou orientação sexual do indivíduo. Nota-se que há grande convergência com a LGPD, mas o GDPR explicita com maior detalhe os dados tecnológicos e digitais, refletindo o ambiente europeu de proteção mais abrangente (EUROPEAN UNION, 2016).

Na prática, enquanto os dados pessoais podem ser tratados sob hipóteses legais mais amplas, como consentimento, execução de contrato ou cumprimento de obrigação legal, os dados sensíveis exigem hipóteses mais restritivas e maior diligência por parte das organizações. Por exemplo, informações de contato de um cliente (dados pessoais) podem ser utilizadas para envio de uma fatura, enquanto dados de saúde (dados sensíveis) só podem ser tratados em situações específicas e com mecanismos de proteção reforçados.

Portanto, compreender a distinção entre dados pessoais e sensíveis, bem como as diferenças entre legislações como a LGPD e o GDPR, é essencial para a conformidade legal e para a adoção de boas práticas em segurança da informação, principalmente em sistemas de comunicação e *mensageria*, nos quais circulam informações de diferentes naturezas.

Além da LGPD, o cenário regulatório brasileiro vem sendo constantemente atualizado pela Autoridade Nacional de Proteção de Dados (ANPD), responsável por estabelecer diretrizes complementares para o tratamento seguro de dados pessoais. Entre as regulamentações recentes, destacam-se as resoluções relacionadas à transferência internacional de dados pessoais, cláusulas-padrão contratuais e à agenda regulatória para o biênio 2025/2026, reforçando a necessidade de mecanismos robustos de proteção em ambientes distribuídos e sistemas de *mensageria*. Essas atualizações demonstram que a conformidade regulatória deixou de ser apenas uma preocupação jurídica, passando a representar também um requisito técnico para arquiteturas modernas de comunicação assíncrona (ANPD, 2026).

3. METODOLOGIA

Este capítulo descreve o método utilizado para o desenvolvimento do presente trabalho, que visa garantir a privacidade e integridade em protocolos de *mensageria* assíncrona. A metodologia adotada tem como base a pesquisa exploratória e bibliográfica, com foco na análise teórica de práticas e ferramentas relacionadas à segurança da comunicação assíncrona. A seguir, detalha-se cada aspecto envolvido.

O objetivo da pesquisa é analisar como as estratégias de segurança aplicadas em protocolos de *mensageria* assíncrona garantem a privacidade e a integridade das mensagens, com ênfase em sistemas amplamente utilizados como Kafka e RabbitMQ. A pesquisa também busca identificar as vulnerabilidades mais comuns e os mecanismos eficazes para mitigá-las.

A abordagem adotada para o desenvolvimento deste trabalho é qualitativa, uma vez que o foco está na análise detalhada de conceitos, práticas e mecanismos de segurança, sem a coleta de dados empíricos. A pesquisa qualitativa permite explorar em profundidade os diferentes aspectos relacionados à segurança em *mensageria* assíncrona e os desafios associados à privacidade e integridade de dados.

A pesquisa é do tipo exploratória, pois visa proporcionar uma compreensão inicial e aprofundada sobre o tema, levantando aspectos fundamentais sobre a segurança em sistemas de *mensageria* assíncrona. A pesquisa exploratória foi escolhida devido ao seu caráter de investigação sobre um tema que, embora presente na literatura, demanda maior compreensão em termos de desafios de segurança.

Além da abordagem exploratória e bibliográfica, este trabalho também pode ser caracterizado como uma pesquisa experimental aplicada, uma vez que envolve a criação de um ambiente controlado para simulação de cenários reais de comunicação assíncrona utilizando Apache Kafka e RabbitMQ. A etapa experimental permitiu validar, na prática, os mecanismos de segurança estudados teoricamente, possibilitando a análise de comportamentos relacionados à criptografia, autenticação, autorização e isolamento de rede em ambientes distribuídos.

Também serão explorados manuais técnicos de ferramentas de *mensageria* para complementar a análise.

A amostra do estudo será delimitada por protocolos de *mensageria* amplamente adotados em ambientes de comunicação assíncrona, com foco em sistemas distribuídos que utilizam Kafka e MQTT. A análise será centrada nas práticas de segurança voltadas à privacidade e integridade de dados em sistemas críticos, como plataformas de pagamento, ambientes IoT e serviços de streaming. As práticas adotadas em ambientes corporativos e industriais também serão consideradas para identificar possíveis variações nos mecanismos de segurança.

Os dados coletados na revisão bibliográfica serão organizados e categorizados por temas, como criptografia, assinatura digital, políticas de entrega e análise de vulnerabilidades e suas mitigações. A tabulação será realizada manualmente, com a aplicação de técnicas de análise qualitativa para identificar padrões, semelhanças e diferenças entre as soluções apresentadas na literatura.

Serão o foco principal, uma vez que a prioridade é o aspecto da segurança.

Com essa metodologia, espera-se atingir os objetivos propostos e fornecer uma contribuição significativa para a área de segurança da informação aplicada a sistemas de comunicação assíncrona.

3.1. Etapas da pesquisa

O desenvolvimento da pesquisa foi dividido em etapas sequenciais. Inicialmente, realizou-se uma revisão bibliográfica sobre comunicação assíncrona, sistemas de troca de mensagens e salvaguardas para a transmissão de dados sigilosos. Em seguida, examinaram-se as principais fragilidades em ambientes distribuídos, atentando para riscos como a interceptação de mensagens, acessos indevidos e erros de configuração.

Depois, montou-se um ambiente experimental com containers Docker para emular arquiteturas de comunicação assíncrona utilizando Apache Kafka e RabbitMQ. Uma vez montado o ambiente, implementaram-se medidas de segurança como criptografia TLS, autenticação, gestão de permissões e segregação de rede.

Por fim, foram realizados testes de validação com diferentes cenários de comunicação, abrangendo tentativas de acesso sem credenciais, envio de mensagens codificadas e avaliação das autorizações concedidas aos usuários, o que possibilitou confirmar a eficiência das proteções de segurança estabelecidas.

4. DESENVOLVIMENTO

Neste capítulo, serão apresentados os aspectos práticos relacionados à segurança na transmissão de dados pessoais e sensíveis em sistemas de *mensageria*. Inicialmente, será discutida a problemática envolvendo vulnerabilidades presentes em ambientes distribuídos e os riscos associados à ausência de mecanismos adequados de proteção. Em seguida, serão abordadas as principais tecnologias utilizadas no estudo, com foco nas plataformas Apache Kafka e RabbitMQ, destacando suas características e aplicações em arquiteturas modernas. Posteriormente, serão apresentados os conceitos de segurança aplicados à *mensageria*, incluindo criptografia, autenticação, autorização, integridade das mensagens e principais vulnerabilidades. Por fim, será descrita a implementação prática do ambiente de testes, contemplando as configurações de segurança aplicadas, os testes realizados e os ganhos obtidos com a utilização dos mecanismos de proteção adotados.

4.1. Problemática

A segurança em aplicações é um dos pilares importante no desenvolvimento de *software*, visto que toda aplicação deve ser resiliente às ameaças externas, com foco maior em segurança quando se trata de uma aplicação comercial e/ou quando se trata de processos bancários ou informações sensíveis de clientes. De forma diferentes, todos os sistemas, independente da arquitetura usada, deve ter preocupações com seguranças a serem abordadas, claro que em níveis diferentes, por exemplo: um sistema baseado na arquitetura micro-serviço, é muito mais propenso a sofrer com alguma ameaça, visto que sua infraestrutura aborda diversos pontos distribuídos de conexão ou integração, consequentemente diversos pontos de possíveis falhas, logo temos um nível de preocupação mais elevada. Já um sistema baseado na arquitetura monolítica, quase todo o processo ocorre dentro de sua própria infraestrutura, pois não é necessário se conectar a outra instância para realizar ações, ou seja, todos as ações que ele realiza, está contido dentro de suas próprias

“fronteiras”, e consequentemente temos menor quantidade de pontos de falhas comparado a arquitetura anterior (KLEPPMANN, 2017).

Toda a escolha de qual arquitetura seguir, seja ela mais segura ou não, deve ser feito de acordo com o cenário em cada empresa, e todas tem seus pontos positivos e negativos, como exemplo de uma arquitetura monolítica, o qual tem uma menor quantidade de pontos de falhas, aumentando a segurança, desempenho, simplicidade no desenvolvimento e também custos mais baixos, porém isso tudo vai em contra a resiliência operacional. Em caso de uma falha de infraestrutura, todos os serviços contidos naquela aplicação ficarão indisponíveis, já que estão todos em uma única estrutura de arquitetura monolítica e servidor único. O oposto da arquitetura de microsserviços, o qual podem estar em servidores diferentes e garantir que outros serviços continuem funcionando em caso de falha da infraestrutura.

Visto essa situação em que qualquer escolha feita baseada no cenário sistêmico corporativo, deverá levar em conta o ganho e o detrimento de algumas características, surgiram algumas ferramentas da comunidade onde o intuíto é categorizar esses riscos de segurança em softwares e tentar diminuí-los, e uma dessas ferramentas é chamada OWASP Top 10, um documento de conscientização para desenvolvedores focado em riscos de segurança mais críticos, apresentados como: “Globalmente reconhecido pelos desenvolvedores como o primeiro passo para uma codificação mais segura.”, onde são um compilado de relatos de entidades e organizações parceiras que o apoiam.

Exemplos dos temas catalogados pelo OWASP TOP 10 são: “*cryptography failure*”, “*insecure design*”, ” *software and data integrity failures*”.

Cryptography failure: Os algoritmos criptográficos são fundamentais para garantir a privacidade e a segurança dos dados; no entanto, eles podem ser bastante suscetíveis a falhas de implementação ou configuração. As falhas criptográficas abrangem erros na aplicação da criptografia, configurações inadequadas de algoritmos criptográficos e gerenciamento de chaves de forma não segura. Por exemplo, uma empresa pode empregar um algoritmo hash não seguro para guardar senhas, deixar de adicionar sal às senhas ou utilizar o mesmo sal para todas as senhas dos usuários armazenadas.

Insecure design: Durante o processo de desenvolvimento de *software*, a vulnerabilidade pode ser incorporada de duas formas distintas. Apesar de a maioria

das vulnerabilidades na lista do Top 10 do OWASP serem relacionadas a erros de implementação, esta vulnerabilidade aponta falhas de design que comprometem a segurança do sistema. Por exemplo, se o design de um aplicativo que armazena e processa dados confidenciais não prevê um sistema de autenticação, mesmo que o software seja implementado de acordo com as especificações, ele continuará sendo inseguro e não protegerá esses dados confidenciais de forma adequada.

Software and data integrity failures: Destaca fragilidades na segurança do *pipeline* DevOps de uma organização e nos processos de atualização de software, semelhantes aos que permitiram o ataque à SolarWinds. Esta categoria de vulnerabilidade abrange a dependência de código de terceiros proveniente de fontes ou repositórios não confiáveis, a falta de proteção no acesso ao *pipeline* de CI/CD e a insuficiência na validação da integridade das atualizações aplicadas automaticamente. Por exemplo, se um invasor conseguir trocar um módulo ou dependência confiável por uma versão alterada ou maliciosa, os aplicativos desenvolvidos com essa dependência poderão rodar código malicioso ou se tornar alvos de exploração (OWASP FOUNDATION, 2021; Check Point, 2025).

O que pode parecer um problema menor de segurança de dados pode frequentemente resultar em grandes vulnerabilidades na implementação do Kafka ou do RabbitMQ. Ao aprender sobre essas armadilhas comuns e evitá-las ativamente, é possível fortalecer seu *cluster* e prevenir problemas de segurança.

O Apache Kafka foi concebido para simplificar o uso durante o desenvolvimento, e não para garantir segurança em ambientes de produção. Isso implica que a maioria das funcionalidades de segurança essenciais, como a autenticação e a criptografia do cliente Kafka, vêm desativadas por padrão. Um equívoco frequente é implementar o Kafka usando sua configuração padrão, o que permite que qualquer um que consiga se conectar à sua rede tenha acesso. Nesse contexto, qualquer aplicativo ou usuário pode criar, usar ou gerenciar o *cluster* sem necessidade de autenticação ou autorização, o que representa um risco semelhante às falhas de segurança não corrigidas do Apache em ambientes de servidor HTTP. Nesses casos, a ausência de medidas de proteção torna os sistemas vulneráveis à exploração (TUXCARE, 2025; CONFLUENT INC., 2026).

O principal desafio de segurança em aplicações de *mensageria* é garantir que as mensagens frequentemente incluam dados confidenciais que necessitam de criptografia, já que pode se tratar de informações sensíveis e que não podem ser vazadas externamente, sempre mantendo a confiabilidade, integridade e disponibilidade.

4.2. Tecnologias Envolvidas

Nesta análise, focamos no Apache Kafka e no RabbitMQ, dois sistemas de mensagens bem conhecidos em estruturas distribuídas.

O RabbitMQ funciona como um intermediário, seguindo o protocolo AMQP, e organiza as mensagens em filas para garantir que cheguem ao destino certo. Ele é uma ótima escolha quando a prioridade é ter certeza de que as mensagens serão entregues e controlar exatamente como elas viajam (VMWARE, 2026).

Já o Apache Kafka é uma plataforma que lida com muitos dados rapidamente, registrando cada evento. Ao contrário do RabbitMQ, que usa filas, o Kafka trabalha com fluxos de dados, permitindo que várias pessoas acessem as informações ao mesmo tempo (APACHE SOFTWARE FOUNDATION, 2026).

A diferença crucial entre eles está na forma como lidam com as informações:

- RabbitMQ: organiza por meio de filas;
- Kafka: usa o fluxo contínuo de eventos.

Ainda que sejam úteis, ambos podem apresentar vulnerabilidades se não forem configurados de maneira correta, principalmente no que diz respeito à proteção dos dados. Se as configurações padrão forem mantidas, a criptografia, a verificação de identidade e o controle de acesso podem estar ausentes, colocando em risco as informações transmitidas.

A escolha do Apache Kafka e do RabbitMQ ocorreu devido à ampla adoção dessas ferramentas em arquiteturas modernas de sistemas distribuídos e comunicação assíncrona. O Apache Kafka foi selecionado por sua alta escalabilidade, capacidade de processamento em tempo real e forte presença em ambientes corporativos de grande volume de dados, especialmente em arquiteturas orientadas a eventos e microsserviços. Já o RabbitMQ foi escolhido por sua robustez no gerenciamento de filas, suporte ao protocolo AMQP e ampla utilização em aplicações transacionais que exigem confiabilidade na entrega de mensagens.

Além da relevância de mercado, ambas as ferramentas oferecem recursos nativos relacionados à segurança, como suporte a TLS, autenticação e controle de acesso, permitindo a realização de análises comparativas sobre mecanismos de proteção aplicados à transmissão de dados pessoais e sensíveis (KREPS; NARKHEDE; RAO, 2011).

4.3. Segurança em *Mensageria*

Em configurações de sistemas que se distribuem, a proteção nos canais de troca de mensagens assume um papel de suma importância, ainda mais quando informações privadas e de caráter delicado são transmitidas. Tais sistemas funcionam como elos de ligação na comunicação entre diferentes serviços, o que os transforma em alvos cruciais; caso haja falhas, uma grande quantidade de dados pode ser revelada.

Dessa forma, a proteção deve ser implementada considerando todo o ciclo de vida da mensagem, desde o momento em que é criada até ser utilizada, assegurando os pilares essenciais da segurança da informação: confidencialidade, integridade, disponibilidade, autenticidade e não repúdio.

4.3.1. Criptografia em trânsito (TLS/SSL)

Em estruturas distribuídas, a proteção dos sistemas de mensagens se mostra primordial, principalmente quando lidam com a movimentação de dados privados e delicados. Tais sistemas funcionam como canais de comunicação entre diferentes serviços, assumindo um papel central que, se vulnerável, pode vazar um montante considerável de dados.

Assim, a proteção deve ser pensada abrangendo todas as etapas da mensagem, desde a sua criação até o seu uso, assegurando os pilares da segurança da informação: sigilo, integridade, acesso, veracidade e a garantia de que o remetente não pode negar o envio. A criptografia durante o transporte visa proteger os dados enquanto percorrem o caminho entre quem produz, os intermediários e quem recebe. Em sistemas de mensagens, essa proteção é normalmente ativada através dos protocolos TLS (*Transport Layer Security*) ou SSL (*Secure Sockets Layer*).

Sem essa barreira de proteção, as mensagens podem ser capturadas por invasores usando táticas como *Man-in-the-Middle* (MITM), o que possibilita a leitura ou mesmo a mudança dos dados enviados.

No Apache Kafka, ativar o TLS envolve gerar certificados digitais e configurar propriedades como ssl.keystore.location e ssl.truststore.location. No RabbitMQ, o TLS pode ser ativado ao configurar ouvintes seguros na porta 5671.

Implementar essa prática é fundamental para assegurar que dados confidenciais, como informações financeiras ou pessoais, permaneçam seguros durante todo o processo de transmissão (APACHE SOFTWARE FOUNDATION, 2026; VMWARE, 2026).

4.3.2. Criptografia em repouso

A criptografia em repouso refere-se à proteção das mensagens armazenadas em disco nos *brokers*. Tanto o Kafka quanto o RabbitMQ têm a capacidade de manter as mensagens de forma persistente para assegurar a sua durabilidade, o que pode gerar perigos extras se o armazenamento for posto em risco.

Se não houver criptografia, um intruso que consiga entrar no sistema de arquivos tem a capacidade de ver todas as mensagens guardadas. Para suavizar esse perigo, são usadas abordagens como:

- Encriptação do disco (exemplos: LUKS, BitLocker) (TUXCARE, 2025);
- Encriptação no grau do aplicativo;
- Emprego de volumes seguros em *containers* Docker.

No Kafka, essa proteção é, na maioria das vezes, feita no grau do sistema operacional ou por meio de soluções de terceiros, enquanto no RabbitMQ ela pode ser reforçada com extensões de segurança.

4.3.3. Autenticação

A autenticação garante que apenas entidades autorizadas possam acessar o sistema de *mensageria*. Caso contrário, qualquer programa ligado à rede teria a capacidade de criar ou receber mensagens, representando um risco à proteção.

No Kafka, a autenticação se concretiza por meio de opções como:

- SASL (*Simple Authentication and Security Layer*) (CONFLUENT INC., 2026; VMWARE, 2026);
- SCRAM (*Salted Challenge Response Authentication Mechanism*)(CONFLUENT INC., 2026; VMWARE, 2026);
- Certificados digitais (mTLS) (CONFLUENT INC., 2026; VMWARE, 2026).

No RabbitMQ, a autenticação se fundamenta em nomes de usuário e palavras-chave, com a possibilidade de união ao LDAP ou certificados TLS.

A falta de autenticação figura entre os erros mais frequentes em cenários com configuração inadequada, abrindo espaço para entradas não permitidas no conjunto de servidores.

4.3.4. Autorização (Controle de acesso)

Além de autenticar usuários, é crucial não só verificar a identidade dos utilizadores, mas também regular as suas ações dentro do sistema. Essa gestão é efetuada através de regras de permissão.

No Kafka, são empregadas ACLs (*Access Control Lists*) para especificar autorizações, por exemplo:

- Enviar dados para um tópico;
- Receber dados;
- Criar ou apagar tópicos.

No RabbitMQ, o controle ocorre através de autorizações ligadas a utilizadores e servidores virtuais (*vhosts*).

A aplicação do conceito de mínimo privilégio é indispensável, assegurando que cada utilizador possua somente as autorizações indispensáveis (APACHE SOFTWARE FOUNDATION, 2026) .

4.3.5. Integridade das mensagens

A integridade garante que a mensagem não foi alterada durante a transmissão. Podemos confirmar essa garantia através de:

- Certificados digitais exclusivos;
- Códigos de verificação criptografados;
- Conferência de somas de verificação.

No ambiente Kafka, a integridade recebe um suporte parcial do sistema de registro distribuído inerente, embora a aplicação possa reforçar essa segurança com implementações adicionais.

A ausência de uma verificação de integridade abre espaço para invasões, onde as mensagens podem ser alteradas sem que ninguém perceba, pondo em risco a solidez do sistema.

4.3.6. Principais vulnerabilidades em *mensageria*

Diversas vulnerabilidades podem afetar sistemas de *mensageria*, especialmente quando não configurados corretamente. Entre as principais, destacam-se:

- Falta de autenticação;
- Comunicação sem criptografia;
- Exposição de portas na rede;
- Uso de configurações padrão;
- Falhas no gerenciamento de chaves.

Situações que realmente aconteceram revelam que grupos Kafka acessíveis a todos podem liberar o acesso total aos dados, mostrando o quanto é crucial uma configuração segura (OWASP FOUNDATION, 2021; TUXCARE, 2025).

4.4. Prática

Para comprovar as ideias teóricas discutidas neste estudo, foi criado um espaço prático através do uso de containers Docker, visando recriar um ambiente

autêntico de comunicação assíncrona entre sistemas que estão distribuídos, implementando medidas de segurança em serviços de *mensageria*.

4.4.1. Ambiente de Testes

O ambiente foi desenvolvido utilizando a virtualização baseada em containers, por meio do Docker, o que possibilitou a geração de instâncias independentes e de fácil replicação.

Dois ambientes principais foram estabelecidos:

- 1 container com Apache Kafka;
- 1 container com RabbitMQ.

Ambos configurados para funcionar em uma rede separada, assegurando controle sobre o tráfego e viabilizando a aplicação de normas de segurança.

4.4.2. Configuração do Apache Kafka

O Apache Kafka foi implementado com ênfase em segurança e gerenciamento de acesso. As principais configurações realizadas incluíram:

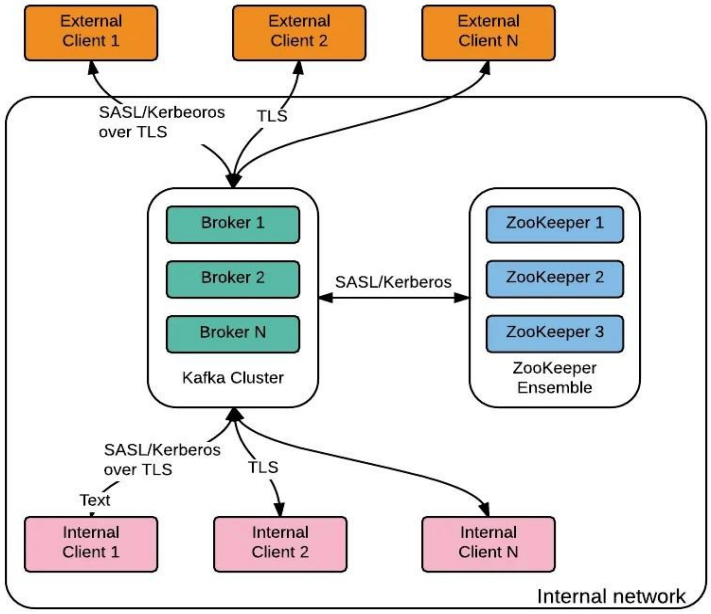
1. Ativação da criptografia TLS

Foi estabelecida uma comunicação segura através de certificados digitais, assegurando que os dados sejam transmitidos de forma criptografada entre os produtores e os consumidores.

- Criação de *keystore* e *truststore;*
- Ajuste de propriedades SSL no *broker*;
- Emprego do protocolo seguro (SSL).

Conforme a Figura 1 utilizada como exemplo, a arquitetura de segurança do Apache Kafka foi configurada utilizando TLS para garantir a criptografia dos dados em trânsito e SASL/Kerberos para autenticação dos clientes e brokers. Dessa forma, além da confidencialidade das mensagens trafegadas, assegura-se que apenas entidades autenticadas possam acessar o cluster Kafka. O ZooKeeper também utiliza autenticação SASL/Kerberos para proteger operações administrativas e metadados do cluster.

Figura 1 - Comunicação segura no Apache Kafka utilizando TLS/SSL e certificados digitais para criptografia dos dados em trânsito.



Fonte: www.automq.com/blog/kafka-security-all-you-need-to-know-and-best-practices (2025).

2. Autenticação com SASL/SCRAM

Foi instaurado o sistema de autenticação por meio de SASL utilizando SCRAM, requerendo credenciais para o acesso ao cluster.

- Cadastro de usuários autenticados;
- Implementação de senhas fortes;
- Limitação de acesso apenas a clientes autorizados.

3. Gestão de acesso (ACLs)

Listas de controle de acesso foram configuradas para restringir as ações disponíveis a cada usuário.

4. Configuração de listeners seguros

O Kafka foi configurado para aceitar conexões apenas via portas seguras, evitando conexões não criptografadas.

4.4.3. Configuração do RabbitMQ

O RabbitMQ foi ajustado com enfoque em segurança, segregação e cifragem.

1. Ativação de TLS

Um *listener* seguro foi implantado utilizando certificados digitais, assegurando que a comunicação seja cifrada.

- Configuração da porta 5671 (TLS) (Figura 2);
- Desabilitação de conexões inseguras (Figura 2);

Figura 2 - Configuração RabbitMQ da porta 5671.

```
# disables non-TLS listeners, only TLS-enabled clients will be able to connect
listeners.tcp = none

listeners.ssl.default = 5671

ssl_options.cacertfile = /path/to/ca_certificate.pem
ssl_options.certfile   = /path/to/server_certificate.pem
ssl_options.keyfile    = /path/to/server_key.pem
ssl_options.verify     = verify_peer
ssl_options.fail_if_no_peer_cert = true
```

Fonte: https://www.rabbitmq.com/docs/ssl (2025).

Conforme a Figura 3, a imagem apresentada é da interface de gerenciamento do RabbitMQ, chamada de RabbitMQ Management Plugin, ela serve para monitorar filas, conexões, portas, *exchanges* e *status* do servidor RabbitMQ.

Ela mostra que:

- O RabbitMQ está aceitando conexões inseguras;
- A porta 5672 está habilitada;
- Não há indicação de TLS/SSL ativo.

Figura 3 - Configuração do RabbitMQ com a porta 5672 insegura ativa.

Fonte: Adaptado da documentação oficial do RabbitMQ (2025).

2. Gestão de usuários

Usuários específicos foram criados com permissões controladas:

- Eliminação do usuário padrão "*guest*";
- Estabelecimento de usuários com senhas robustas;
- Classificação de usuários conforme a função.

3. Controle de acesso por *vhosts*

Virtual hosts (*vhosts*) foram empregados para isolar diferentes ambientes dentro do RabbitMQ (VMWARE, 2026).

- Diferenciação de aplicações por ambiente;
- Limitação de acesso por usuário;

- Isolamento de filas e exchanges.

4. Permissões

   Permissões específicas foram designadas para cada usuário:

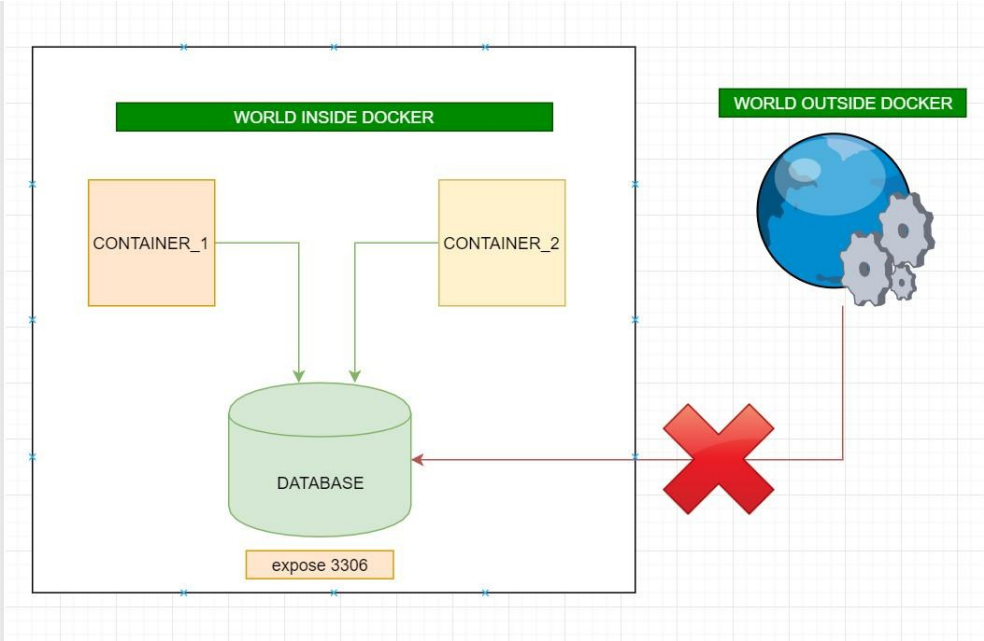
- Leitura (*consume*);
- Escrita (*publish*);
- Administração (*admin*).

4.4.4. Configuração de Rede e Isolamento

A montagem da rede entre os containers foi um aspecto crucial para a proteção do ambiente. Foi implementada uma rede interna do Docker, o que possibilitou a comunicação gerenciada entre os serviços e impediu a exposição direta de portas sensíveis ao ambiente externo. Essa abordagem favoreceu a simulação de um ambiente que se assemelha mais à realidade empresarial, onde o acesso aos sistemas de comunicação é limitado e supervisionado. Ademais, a restrição de acesso ajudou a diminuir a área vulnerável a ataques e a bloquear conexões não permitidas.

A Figura 4 demonstra a segmentação da rede interna Docker utilizada no ambiente. Os *containers* se comunicam de forma isolada dentro da infraestrutura privada, permitindo acesso controlado aos serviços internos, como o banco de dados. O bloqueio de conexões externas diretas reduz a exposição de portas sensíveis e diminui a superfície vulnerável a ataques, reproduzindo práticas comuns de segurança adotadas em ambientes corporativos.

Figura 4 - Isolamento de *containers* e restrição de acesso externo em rede interna Docker.



Fonte: https://stackoverflow.com/questions/40801772/what-is-the-difference-between-ports-and-expose-in-docker-compose (2025).

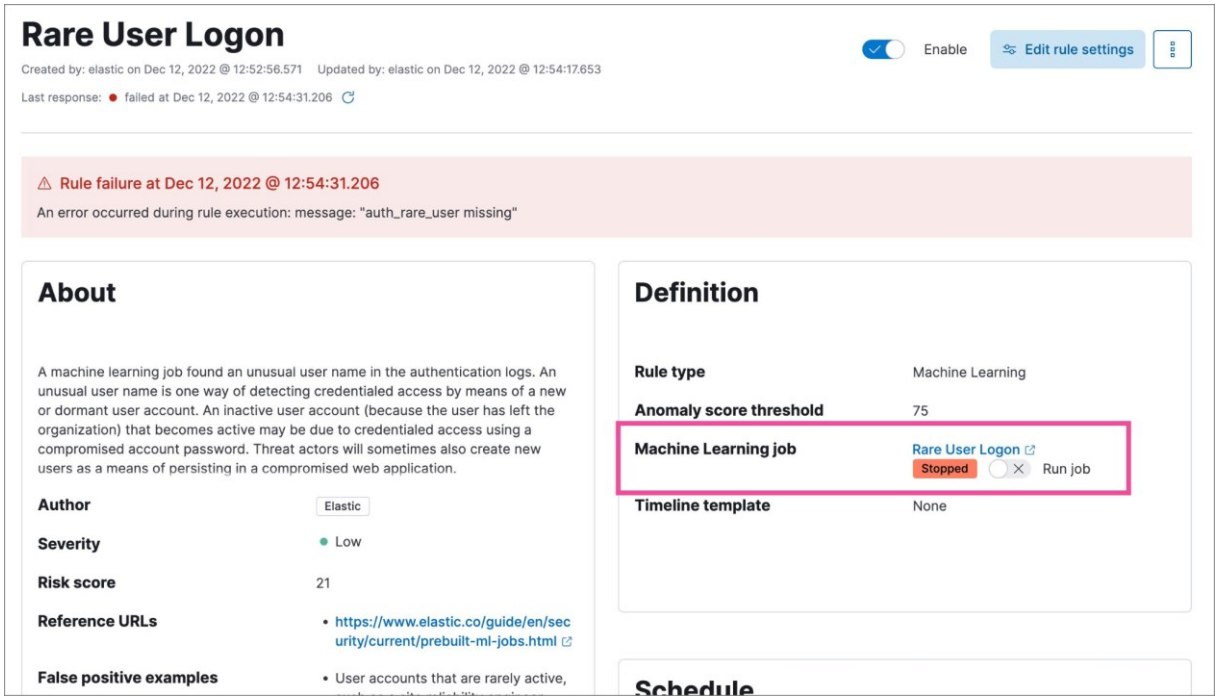
4.4.5. Logs e Monitoramento

Mecanismos de registro de logs foram implementados em ambos os sistemas, proporcionando um acompanhamento minucioso das ações realizadas no ambiente. Esses logs continham dados sobre tentativas de acesso, erros de autenticação e ações realizadas pelos usuários. A geração de logs revelou-se fundamental para auditorias e para reconhecer potenciais incidentes de segurança, além de ajudar na avaliação do comportamento do sistema em várias situações.

A Figura 5 apresenta um exemplo de monitoramento de eventos de autenticação utilizando *Elastic Security*. O sistema analisa *logs* relacionados ao acesso de usuários e utiliza mecanismos de *Machine Learning* para identificar comportamentos incomuns, como *logins* raros ou acessos suspeitos. Esse tipo de monitoramento contribui para auditorias de segurança e para a identificação de

possíveis incidentes relacionados a credenciais comprometidas ou acessos não autorizados.

Figura 5 - Monitoramento de eventos de autenticação e detecção de anomalias utilizando *Elastic Security*.



Fonte: *Elastic Security Documentation* (2025).

4.4.6. Testes Realizados

Para confirmar o ambiente, foram executados testes práticos:

- Envio de mensagens tanto criptografadas quanto não criptografadas;
- Tentativas de acesso sem autenticação;
- Verificação das limitações de ACL;
- Simulação de uso não autorizado.

Os testes mostraram que:

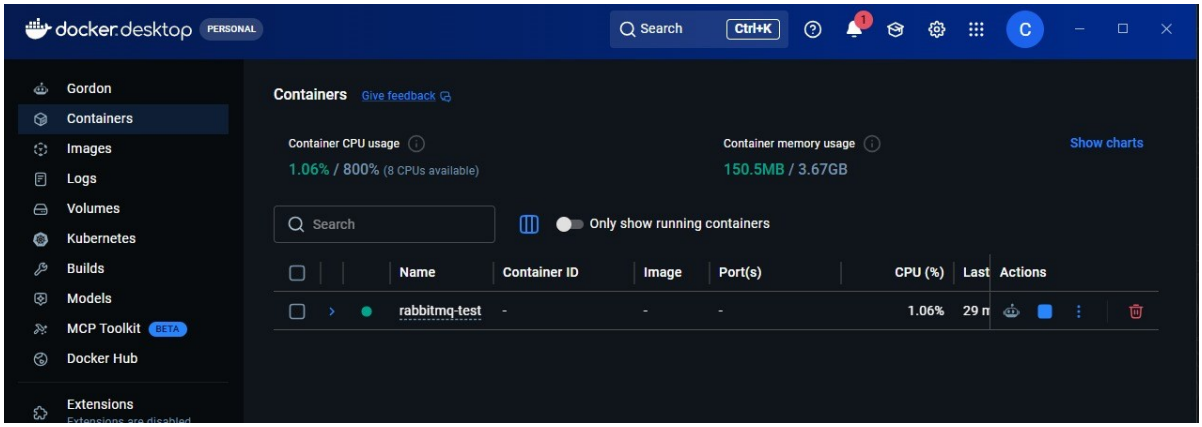
- Sem proteção: acesso completo ao sistema;

- Com proteção: acesso limitado e supervisionado.

Além dos testes funcionais de autenticação e autorização, foram realizados testes de interceptação de tráfego utilizando a ferramenta *Wireshark*, com o objetivo de analisar o comportamento da transmissão de mensagens antes e depois da aplicação de criptografia TLS.

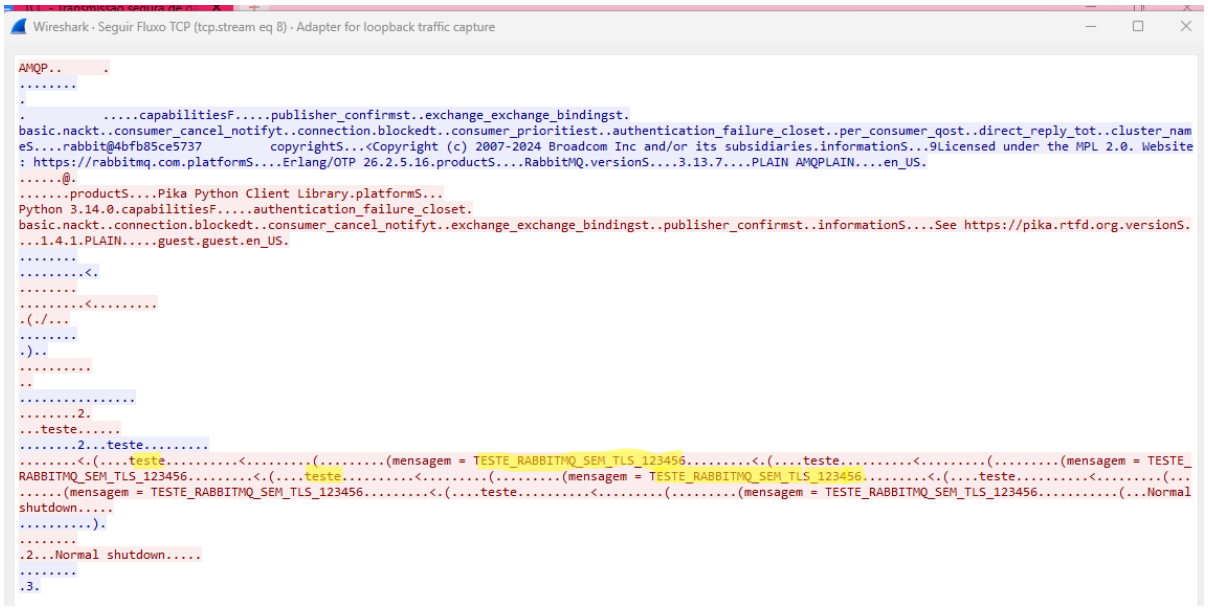
Inicialmente, o ambiente foi executado sem mecanismos de criptografia habilitados, permitindo a captura dos pacotes trafegados entre produtores, brokers e consumidores conforme as Figuras 6 a 10. Nessa etapa, foi possível visualizar o conteúdo das mensagens em texto legível, incluindo informações sensíveis transmitidas pela aplicação, evidenciando a vulnerabilidade do ambiente a ataques do tipo *Man-in-the-Middle* (MITM) (OWASP FOUNDATION, 2021).

Figura 6 - Configuração de ambiente *RabbitMQ* a partir do Docker Desktop.



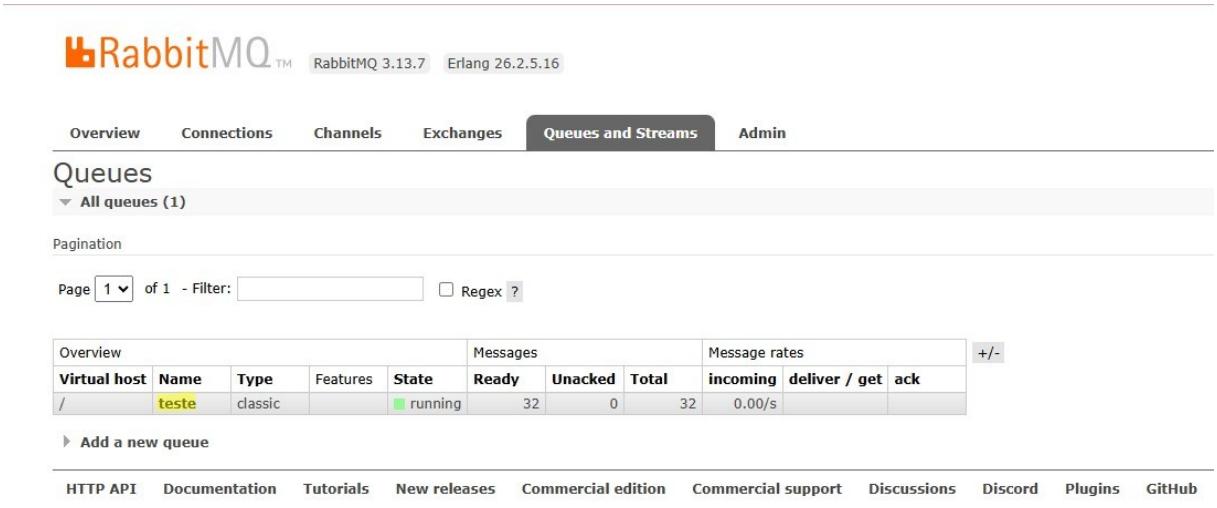
Fonte: Elaborado pelo autor (2026).

Figura 7 - Captura de tráfego no *Wireshark* demonstrando transmissão insegura de dados pelo *RabbitMQ*.



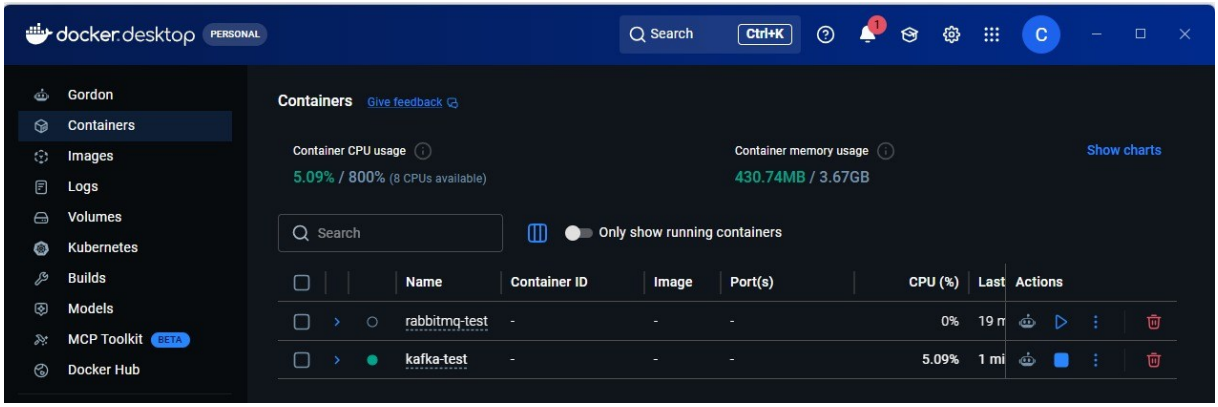
Fonte: Elaborado pelo autor (2026).

Figura 8 - Captura do terminal do *RabbitMQ* demonstrando nome capturado no *Wireshark*.



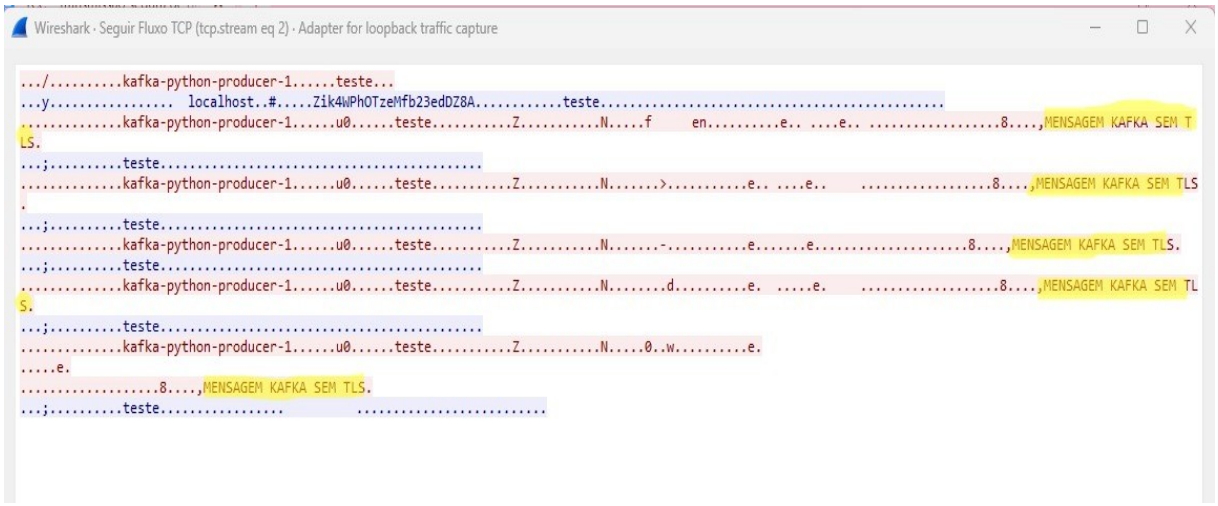
Fonte: Elaborado pelo autor (2026).

Figura 9 - Configuração de ambiente *Kafka* a partir do Docker Desktop.



Fonte: Elaborado pelo autor (2026).

Figura 10 - Captura de tráfego no *Wireshark* demonstrando transmissão insegura de dados pelo *Kafka*.



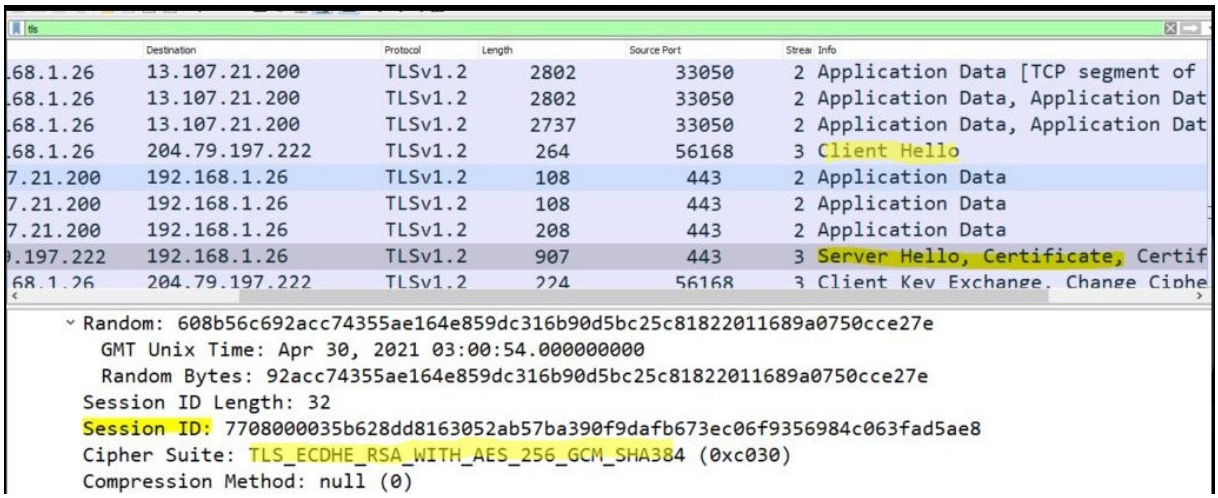
Fonte: Elaborado pelo autor (2026).

Posteriormente, após a implementação do TLS/mTLS nos serviços Apache Kafka e RabbitMQ, novos testes de captura foram realizados. Os resultados demonstraram que os pacotes passaram a trafegar de forma criptografada, impossibilitando a leitura direta do conteúdo das mensagens interceptadas. Dessa forma, verificou-se na prática a efetividade dos mecanismos de segurança aplicados na proteção dos dados em trânsito (CONFLUENT INC., 2026).

A Figura 11 apresenta uma captura de tráfego realizada no Wireshark durante testes de comunicação protegida por TLS 1.2. É possível observar etapas do handshake TLS, como *“Client Hello”* e *“Server Hello”*, além da troca de certificados

digitais para autenticação segura da conexão. Após o estabelecimento da sessão segura, os pacotes passam a ser identificados como *“Application Data”*, indicando que o conteúdo transmitido encontra-se criptografado e não pode ser interpretado diretamente por terceiros que interceptam o tráfego.

Figura 11 - Captura de tráfego TLS 1.2 no Wireshark demonstrando *handshake* criptográfico e transmissão segura de dados.



Fonte: Adaptado de *Wireshark Documentation* (2025).

4.4.7. Disponibilização do Ambiente

O ambiente criado pode ser acessado através de imagens Docker, possibilitando sua cópia em diversos cenários. Essa estratégia torna mais fácil a reutilização das configurações aplicadas e ajuda na uniformização de ambientes seguros. Ademais, a disponibilização em serviços como Docker Hub permite que outros usuários experimentem e verifiquem as soluções sugeridas.

4.5. Ganhos de Segurança

A aplicação das definições de segurança em ambientes de mensagens resultou em melhorias notáveis na proteção dos dados em trânsito, principalmente no que diz respeito à confidencialidade, integridade e controle de acesso. Em um primeiro momento, ao analisarem configurações padrão, foram identificadas falhas críticas, como a falta de autenticação, comunicação sem criptografia e permissões excessivas, que permitiam o acesso livre a dados transmitidos. Com a adoção das medidas sugeridas, esses perigos foram significativamente diminuídos, demonstrando a eficácia das práticas implementadas (OWASP FOUNDATION, 2021).

No que tange à confidencialidade, a implementação de criptografia durante a transmissão via TLS assegurou que as mensagens não fossem capturadas ou compreendidas por terceiros no momento da comunicação entre produtores, *brokers* e consumidores. Esse recurso se mostrou crucial para proteger dados pessoais e sensíveis, especialmente em situações que envolvem a troca de informações financeiras, cadastrais ou de identificação. Adicionalmente, a criptografia dos dados armazenados melhorou a segurança das informações no armazenamento, evitando que acessos não autorizados aos sistemas de arquivos resultam na revelação de dados críticos (CONFLUENT INC., 2026; OWASP FOUNDATION, 2021; VMWARE, 2026).

Em relação à integridade dos dados, a utilização de protocolos seguros e sistemas de validação garantiu que as mensagens permanecessem inalteradas durante o trânsito. Este fator é vital em sistemas distribuídos, onde diversas operações dependem da precisão das informações recebidas. A manutenção da integridade ajuda a prevenir inconsistências, falhas no processamento e possíveis fraudes geradas pela manipulação inadequada de mensagens (KLEPPMANN, 2017).

Um outro benefício significativo foi a implementação de um controle de acesso mais rigoroso, por meio da autenticação e autorização. A necessidade de credenciais adequadas para acessar os sistemas, combinada com a atribuição de permissões específicas a cada usuário, permitiu limitar as ações realizadas dentro do ambiente. Assim, apenas indivíduos autorizados tiveram a capacidade de produzir ou consumir mensagens, diminuindo consideravelmente o risco de acessos não autorizados. O uso do princípio do menor privilégio ajudou a restringir o impacto de potenciais

vazamentos de credenciais, já que os usuários passaram a acessar apenas os recursos essenciais (CONFLUENT INC., 2026; VMWARE, 2026).

Além dos aspectos de segurança direto, a ativação de registros e auditorias melhorou a visibilidade sobre o funcionamento do sistema. Foi possível documentar e analisar tentativas de acesso, falhas de autenticação e ações realizadas, o que facilitou a identificação de comportamentos suspeitos e uma resposta ágil a incidentes. Esse monitoramento contínuo é fundamental para os ambientes que precisam se manter em conformidade com legislações como a LGPD, permitindo o rastreamento do uso de dados e demonstrando a responsabilidade no tratamento dessas informações (OWASP FOUNDATION, 2021).

Em termos de desempenho, verificou-se que a implementação das camadas de segurança trouxe um leve aumento na latência das comunicações, especialmente por conta do uso de criptografia. Contudo, este impacto foi considerado aceitável em relação aos benefícios obtidos em proteção de dados. No contexto do Apache Kafka, a arquitetura distribuída possibilitou a manutenção de altos níveis de *throughput*, mesmo com as configurações de segurança em funcionamento. Por outro lado, o RabbitMQ mostrou um desempenho estável, apresentando um leve impacto em situações de alta carga, mas continuou eficiente em aplicações transacionais.

Um ponto importante diz respeito à simplicidade na configuração das ferramentas. O RabbitMQ mostrou ser mais fácil na instalação de mecanismos de segurança, sendo mais apropriado para situações menos complexas. Em contraste, o Apache Kafka, apesar de exigir uma configuração mais complicada, revelou-se mais forte e escalável, sendo ideal para cenários que precisam de grande volume de dados e processamento contínuo.

Dessa forma, os resultados obtidos evidenciam que a aplicação de práticas de segurança em sistemas de *mensageria* é fundamental para garantir a proteção de dados pessoais e sensíveis. A utilização de métodos como criptografia, autenticação, controle de acesso e auditoria não só diminui as vulnerabilidades, mas também eleva a confiabilidade e a resistência dos sistemas. A seleção da ferramenta ideal deve levar em conta não apenas fatores de desempenho, mas também os requisitos de

segurança e o contexto em que será utilizada, especialmente em ambientes que manipulam informações críticas.

5. CONCLUSÃO

O objetivo deste estudo foi investigar e sugerir métodos para a transmissão segura de informações pessoais e sensíveis em sistemas que utilizam comunicação assíncrona, focando especialmente em plataformas de *mensageria* como Apache Kafka e RabbitMQ.

Durante a pesquisa, ficou claro que, apesar das vantagens notáveis da comunicação assíncrona em termos de escalabilidade e resiliência, também existem desafios significativos que envolvem a segurança da informação. A falta de mecanismos apropriados pode fazer com que os sistemas fiquem vulneráveis a riscos como interceptação, vazamento e alteração de dados (REDDI; NARAYANAN, 2021).

A investigação das tecnologias mostrou que tanto o Kafka quanto o RabbitMQ são soluções robustas, mas requerem configurações adicionais para assegurar a segurança adequadamente. A adoção de procedimentos como criptografia, autenticação, controle de acesso e auditoria é fundamental para salvaguardar os dados enquanto estão em trânsito e em repouso.

Adicionalmente, a execução prática em ambientes simulados ajudou a confirmar a relevância da configuração segura dessas ferramentas, demonstrando melhorias significativas na proteção das informações, com pouco impacto na performance.

As limitações do estudo incluem a falta de testes em ambientes reais de produção e a análise restrita a apenas duas ferramentas de *mensageria*. Para investigações futuras, recomenda-se considerar abordagens como:

- Arquiteturas *Zero Trust*;
- Integração com tecnologia de *blockchain*;
- Uso de criptografia avançada e *mensageria* quântica.

Conclui-se que a segurança em sistemas de *mensageria* é uma necessidade não apenas técnica, mas um elemento crucial para assegurar confiança, privacidade e conformidade com regulamentações como a LGPD.

Além disso, percebe-se que a efetividade dessas ações está intimamente ligada à maneira como são colocadas em prática, demonstrando que as configurações padrão não atendem adequadamente ambientes que manipulam informações sensíveis. A implementação correta de boas práticas de segurança não só diminui as vulnerabilidades, mas também eleva a confiança e a robustez dos sistemas distribuídos.

Dessa maneira, realça-se a relevância de uma abordagem contínua e adaptativa em segurança da informação, levando em conta o progresso constante das ameaças cibernéticas. Assim, empresas que adotam medidas preventivas e investem na segurança de seus sistemas de comunicação encontram-se em uma posição mais favorável para assegurar a proteção de dados, a conformidade com regulamentos e a confiança de seus usuários em um ambiente que se torna cada vez mais digital e interconectado.

Apesar dos ganhos significativos obtidos com a implementação de mecanismos como TLS, autenticação e controle de acesso, é importante destacar que a segurança em sistemas de *mensageria* não depende exclusivamente da aplicação de tecnologias criptográficas. Medidas como TLS/mTLS protegem os dados durante o trânsito, mas não eliminam riscos relacionados à má gestão de credenciais, comprometimento de chaves privadas, configurações inadequadas de permissões ou falhas operacionais. Dessa forma, a segurança deve ser tratada como um processo contínuo, envolvendo não apenas soluções técnicas, mas também políticas de governança, monitoramento, auditoria e gestão segura de certificados digitais e chaves criptográficas.

6. REFERÊNCIAS

AUTOMQ TEAM. *Kafka Security: All You Need to Know & Best Practices*. Disponível em: https://www.automq.com/blog/kafka-security-all-you-need-to-know-and-best-practices. Acesso em: 01 mai. 2026.

AUTORIDADE NACIONAL DE PROTEÇÃO DE DADOS (ANPD). *Regulamentações da ANPD*. Disponível em: https://www.gov.br/anpd/pt-br/acesso-a-informacao/institucional/atos-normativos/regulamentacoes_anpd/regulamentacoes-anpd. Acesso em: 14 abr. 2026.

APACHE SOFTWARE FOUNDATION. *Apache Kafka Documentation*. Disponível em: https://kafka.apache.org/documentation/. Acesso em: 14 mar. 2026.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. *Lei Geral de Proteção de Dados Pessoais (LGPD)*. Diário Oficial da União: seção 1, Brasília, DF, 15 ago. 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 14 mar. 2026.

CHECK POINT. *OWASP Top 10 aplicativos de vulnerabilidade de segurança da web*. Disponível em: https://www.checkpoint.com/pt/cyber-hub/cloud-security/what-is-application-security-appsec/owasp-top-10-vulnerabilities. Acesso em: 06 nov. 2025.

CONFLUENT INC. *Apache Kafka Documentation*. Disponível em: https://kafka.apache.org/documentation/. Acesso em: 14 abr. 2025.

CORRÊA, Iago. *Proposta de configuração para garantia de entrega de mensagens no Apache Kafka*. Disponível em: https://repositorio.ufsm.br/bitstream/handle/1/25169/DIS_PPGCC_2022_CORR%C3%8AA_IAGO.pdf?sequence=1&isAllowed=y. Acesso em: 02 dez. 2024.

ELASTIC. *Elastic Security*. Disponível em: https://www.elastic.co/pt/observability/infrastructure-monitoring. Acesso em: 25 mai. 2026.

EUROPEAN UNION. *General Data Protection Regulation (GDPR): Regulation (EU) 2016/679 of the European Parliament and of the Council*. Brussels, 2016.

KLEPPMANN, Martin. *Designing Data-Intensive Applications*. Sebastopol: O'Reilly Media, 2017.

KREPS, Jay; NARKHEDE, Neha; RAO, Jun. *Kafka: A Distributed Messaging System for Log Processing*. [S.l.], 2011.

OWASP FOUNDATION. *OWASP Top 10: 2021*. Disponível em: https://owasp.org/www-project-top-ten/. Acesso em: 14 abr. 2025.

RABBITMQ. *TLS Support*. Disponível em: https://www.rabbitmq.com/docs/ssl. Acesso em: 01 mai. 2026.

REDDI, S.; NARAYANAN, P. *Building Scalable Systems with Asynchronous Messaging*. [S.l.]: Pearson Publishing, 2021.

RESULTADOS DIGITAIS. *Comunicação assíncrona: o que é, boas práticas e ferramentas*. Disponível em: https://www.rdstation.com/blog/marketing/comunicacao-assincrona/. Acesso em: 09 jun. 2024.

STACK OVERFLOW. *What is the Difference Between Ports and Expose in Docker-Compose?*. Disponível em: https://stackoverflow.com/questions/40801772/what-is-the-difference-between-ports-and-expose-in-docker-compose. Acesso em: 01 mai. 2026.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. *Sistemas distribuídos: princípios e paradigmas*. 2. ed. São Paulo: Pearson Prentice Hall, 2019.

TUXCARE. *2025 Apache Kafka Security: Best Practices & EOL Risks*. Disponível em: https://tuxcare.com/blog/apache-kafka-security. Acesso em: 06 nov. 2025.

VMWARE. *RabbitMQ Documentation*. Disponível em: https://www.rabbitmq.com/documentation.html. Acesso em: 14 abr. 2026.

ZENDESK. *Comunicação síncrona e assíncrona: como trabalhar com seu time*. Disponível em: https://www.zendesk.com.br/blog/comunicacao-sincrona-assincrona/. Acesso em: 08 mar. 2024.