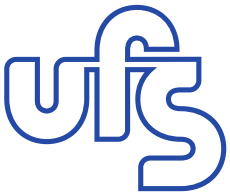


UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Análise de Superfície de Ataques Aplicada a Microsserviço

Dissertação de Mestrado

Tacyanne Bernadete Lima Pimentel

São Cristóvão – Sergipe

2026

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Tacyanne Bernadete Lima Pimentel

Análise de Superfície de Ataques Aplicada a Microsserviço

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Fabio Gomes Rocha

São Cristóvão – Sergipe

2026

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Pimentel, Tacyanne Bernadette Lima
N518a Análise de superfície de ataques aplicada a microsserviço / Tacyanne Bernadette Lima Pimentel ; orientador Fabio Gomes Rocha. - São Cristóvão, 2025.
101 f.; il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2025.

1. Computadores – Medidas de segurança. 2. Software - Proteção. 3. Engenharia de software. 4. Sistemas operacionais distribuídos (Computadores) I. Rocha, Fabio Gomes orient. II. Título.

CDU 004.5

Agradecimentos

Este trabalho é dedicado à minha mãe, que, embora não esteja mais presente, permanece como a força motriz de minha existência, a essência do amor e a personificação do ser humano. A ela, que me transmitiu os fundamentos da vida e da humanidade, a razão de minha jornada.

À minha irmã, meu pilar de apoio inabalável. Mesmo diante das minhas próprias incertezas, ela deposita total confiança no meu potencial, fazendo-me sentir única e especial. É uma honra poder ouvir minha frase predileta: “Tacynha, não ria”, um ensinamento que busco internalizar, embora o desafio persista. Sua perspectiva me concede um senso de singularidade e a satisfação, de algum modo, de poder contribuir para a grandiosidade que ela representa para mim.

Ao meu pai, companheiro de aventuras e mentor na busca por melhores leituras e decisões que tomo em minha vida. Reconheço que, na minha inexperiência, eu nem sempre enxergo as coisas com a clareza dele, e suas ideias, por vezes, parecem distantes, mas invariavelmente moldam minha realidade.

Às minhas três tias: à **Tia Deia**, um pilar de apoio incondicional tanto para os sonhos que já reconheço quanto para aqueles que ainda não consigo mensurar; à **Tia Aguinha**, que me impulsiona com esperança, inundando-me de desejos e vontades para me lançar a novos desafios; e à **Tia Danda**, cujas teorias acendem a fé de que Nanynha me guia, me escuta e me faz compreender que o mundo transcende minha capacidade de entendimento e interpretação. Em suas distintas particularidades, elas perpetuam o conceito de maternidade, moldando o ser humano que me tornei.

Aos meus primos e demais familiares, que compõem a essência de quem sou e represento.

Resumo

Contexto: A arquitetura de microsserviços *(MSA)* tornou-se proeminente no desenvolvimento de sistemas distribuídos por promover escalabilidade, modularidade e independência ao fragmentar aplicações monolíticas em serviços menores e coesos, que se comunicam via interfaces definidas (*APIs REST* ou sistemas de mensagens). Contudo, a adoção da *(MSA)* introduz complexidades operacionais e técnicas notáveis, como a gestão de falhas, a complexidade inerente à comunicação e a observabilidade em ambientes distribuídos, além de lidar com a consistência eventual dos dados. **Método**: Esta dissertação adota a metodologia *Design Science Research (DSR)* , estruturada nas três etapas do seu ciclo: aquisição de conhecimento, construção do modelo e avaliação. A fase de aquisição envolveu um Mapeamento Sistemático de Literatura *(MSL)* para identificar práticas, lacunas e desafios em segurança e requisitos não funcionais em microsserviços. Esse *MSL* serviu de base para a construção do modelo proposto, ancorado nas técnicas *EventStorming* e *Behavior-Driven Development (BDD)*, com foco na análise da superfície de ataque e na verificação sistemática dos requisitos. **Resultados**: A aplicação da proposta metodológica demonstrou eficácia na identificação e análise da superfície de ataque em microsserviços. A técnica de *EventStorming* viabilizou o mapeamento de vulnerabilidades e pontos críticos de exposição, promovendo melhorias na visibilidade dos processos e na colaboração entre equipes. Adicionalmente, o estudo de caso com *BDD* permitiu a verificação sistemática de requisitos não funcionais (segurança e desempenho), resultando em benefícios expressivos na automação de testes e na rastreabilidade dos requisitos, ampliando a confiabilidade do sistema. Em síntese, a solução é aplicável, escalável e contribui para o fortalecimento da engenharia de software orientada à segurança em ambientes distribuídos.

Palavras-chave: Arquitetura de Microsserviços *(MSA)*; Segurança de Software; Requisitos Não Funcionais; Superfície de Ataque; *EventStorming*; *Behavior-Driven Development (BDD)*; *Design Science Research (DSR)*; Engenharia de Software Segura; Sistemas Distribuídos; Complexidade Operacional; Consistência Eventual.

Abstract

Context: Microservices Architecture (*MSA*) has become prominent in the development of distributed systems by promoting scalability, modularity, and independence by fragmenting monolithic applications into smaller, cohesive services that communicate via defined interfaces (*REST APIs* or messaging systems). However, the adoption of *MSA* introduces notable operational and technical complexities, such as fault management, the inherent complexity of communication, and observability in distributed environments, in addition to dealing with eventual data consistency. **Method**: This dissertation adopts the *Design Science Research (DSR)* methodology, structured into the three stages of its cycle: knowledge acquisition, model construction, and evaluation. The acquisition phase involved a Systematic Literature Mapping (*SLM*) to identify practices, gaps, and challenges in security and non-functional requirements in microservices. This *SLM* served as the basis for the construction of the proposed model, anchored in the ***EventStorming* and *Behavior-Driven Development (BDD)*** techniques, focusing on the analysis of the attack surface and the systematic verification of requirements. **Results**: The application of the proposed methodology demonstrated effectiveness in identifying and analyzing the attack surface in microservices. The *EventStorming* technique enabled the mapping of vulnerabilities and critical exposure points, promoting improvements in process visibility and collaboration between teams. Additionally, the case study with *BDD* allowed for the systematic verification of non-functional requirements (security and performance), resulting in expressive benefits in test automation and requirements traceability, increasing system reliability. In summary, the solution is applicable, scalable, and contributes to strengthening security-oriented software engineering in distributed environments.

Keywords: Microservices Architecture (*MSA*); Software Security; Non-Functional Requirements; Attack Surface; *EventStorming*; *Behavior-Driven Development (BDD)*; *Design Science Research (DSR)*; Secure Software Engineering; Distributed Systems; Operational Complexity; Eventual Consistency.

Lista de ilustrações

Lista de abreviaturas e siglas

ADS	Ambiente de Desenvolvimento de Software
ASVS	Application Security Verification Standard
BDD	Behavior-Driven Development
BSIMM	Building Security In Maturity Model
DDD	Domain-Driven Design
DF	Diagramas de Fluxo de Dados
EDA	Arquitetura Orientada a Eventos
IAST	Interactive Application Security Testing
ICT	Information and Communication Technology
ISO/IEC	International Organization for Standardization e da IEC – International Electrotechnical Commission
LLMs	Large Language Models
ML	Machine Learning
MSA	Arquitetura de Microsserviços
MSL	Mapeamento Sistemático de Literatura
NIST	Instituto Nacional de Padrões e Tecnologia
OWASP	Open Worldwide Application Security Project
PCI	Security Standards Council
QA	Quality Assurance
SAMM	Security Assurance Maturity Model
SLM	Systematic Literature Mapping
SSDLC	Ciclo de Vida de Desenvolvimento de Software Seguro
TDD	Test-Driven Development
TOGAF	The Open Group Architecture Framework

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS. Candidato: Tacyanne Bernadete Lima Pimentel

Em 28 dias do mês de Agosto do ano de dois mil e vinte cinco, com início às 20hs, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado da candidata **Tacyanne Bernadete Lima Pimentel**, que desenvolveu o trabalho intitulado: **" Análise de Superfície de Ataques Aplicada a Microsserviço",** sob a orientação do Prof. Dr. **Fábio Gomes Rocha.** A Sessão foi presidida pelo Prof. Dr. **Fábio Gomes Rocha** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Guillermo H. Rodríguez** (UBA) e, em seguida, o Prof. Dr. **Gilton José Ferreira da Silva** (Procc/UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) Aprovada "*(aprovado/reprovado)"*. Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 28 de Agosto de 2025.

Prof. Dr. Fábio Gomes Rocha
(PROCC/UFS)
Presidente

Prof. Dr Guillermo H. Rodríguez
(UNICEN)
Examinador Externo

Prof. Dr. Gilton José Ferreira da Silva
(PROCC/UFS)
Examinador Interno

Tacyanne Bernadete Lima Pimentel
Candidato

Sumário

1

Introdução

A evolução do desenvolvimento de software tem sido impulsionada pela busca por maior agilidade, escalabilidade e resiliência. Nesse cenário, o estilo arquitetônico de microsserviços tem se destacado como uma abordagem eficaz para empresas que desenvolvem aplicações web em larga escala, permitindo modificações ágeis e respostas eficientes às dinâmicas do mercado (SCACCHI; ALSPAUGH, 2019)

Paralelamente, a arquitetura orientada a eventos (EDA) (SILVA et al., 2025) tem sido amplamente adotada na construção de aplicativos baseados em eventos, em substituição ao modelo tradicional de solicitação/resposta.

Nesse paradigma, a ocorrência de um evento pode desencadear diversas ações ou processos, tornando as aplicações mais dinâmicas e descentralizadas. Sistemas baseados em EDA demonstram maior agilidade, escalabilidade e capacidade de adaptação, respondendo de forma eficiente às demandas dos negócios digitais. Além disso, a EDA aprimora a segurança ao permitir que cada serviço reaja exclusivamente aos eventos para os quais foi programado, reduzindo a superfície de ataque e minimizando interações não autorizadas.

Sistemas desenvolvidos com base nessa abordagem apresentam maior agilidade, escalabilidade e capacidade de adaptação ao contexto, tornando-se mais responsivos às demandas dos negócios digitais. Além disso, a arquitetura orientada a eventos contribui para o aprimoramento da segurança, uma vez que cada serviço reage exclusivamente aos eventos para os quais foi programado, reduzindo a superfície de ataque e minimizando interações não autorizadas.

A incorporação da segurança no inicio do processo de desenvolvimento de software é crucial e é efetivada por meio do Ciclo de Vida do Desenvolvimento de Software Seguro (SDLC) (RAMIREZ; AIELLO; LINCKE, 2020). Este processo exige ações contínuas e práticas que garantam a confidencialidade, integridade e disponibilidade das aplicações. O aumento da sofisticação e do impacto dos ataques cibernéticos nos últimos anos sublinha a importância de incluir a segurança em todas as etapas do ciclo de vida dos projetos e serviços, especialmente

considerando que a maioria das falhas ocorre durante a fase de desenvolvimento. É fundamental que as equipes compreendam os conceitos de segurança, adotando padrões de mercado e escolhendo soluções corporativas reutilizáveis para um maior controle sobre as atividades de desenvolvimento.

1.1 Justificativa

O *Domain-Driven Design (DDD)* (RADEMACHER; SORGALLA; SACHWEH, 2018) estabelece uma relação sinérgica com a arquitetura de microsserviços (SCHMIDT; THIRY, 2020), favorecendo o desenvolvimento de sistemas escaláveis, flexíveis e de fácil manutenção, capazes de se adaptar às dinâmicas das demandas de negócio. A contínua aprimoração da decomposição de sistemas em microsserviços é impulsionada pela busca por metodologias mais eficientes e inteligíveis para o projeto e a construção de software. Essa evolução engloba a delimitação precisa dos contextos de cada microsserviço e a definição de critérios funcionais para sua organização estrutural. Tais práticas contribuem significativamente para o isolamento de falhas, a simplificação da implantação e a aceleração do ciclo de desenvolvimento, promovendo uma integração eficiente com os princípios do *DevOps* contemporâneo.

Em um cenário de sistemas distribuídos, como os construídos com microsserviços, é imperativo estabelecer camadas de segurança que integrem a infraestrutura e as redes de aplicação. Dada a complexidade inerente a esses sistemas, a atenção aos dados específicos necessários para a implantação bem-sucedida dos componentes é crucial. No contexto dos microsserviços, a modelagem e a transformação dos dados do domínio de negócio são centrais em todo o ciclo de vida do sistema, impactando diretamente a segurança. Para mitigar os desafios de segurança associados a essa arquitetura, os desenvolvedores devem adotar as melhores práticas e tecnologias disponíveis. Isso não só assegura a uniformidade e a capacidade de recuperação frente a falhas, mas também promove a proteção eficaz das informações e a eficiência na comunicação segura entre os componentes.

Diante disso, o objetivo central desta pesquisa é propor e avaliar uma metodologia que aprimore a identificação e validação de requisitos de segurança em sistemas de software, com foco particular na análise da superfície de ataque em arquiteturas de microsserviços. Essa metodologia visa melhorar a capacidade de um sistema de software de resistir a ataques, proteger dados sensíveis e manter a funcionalidade, mesmo sob condições adversas.

Para atingir este propósito, a investigação adotou a abordagem de *Design Science Research (DSR)*, que se desdobrou em três fases interligadas: aquisição de conhecimento, construção do modelo e avaliação. A fase de aquisição de conhecimento consistiu na realização de um Mapeamento Sistemático de Literatura *(MSL)*, cujo intuito foi caracterizar as práticas correntes e identificar as lacunas existentes no que tange à avaliação de requisitos não funcionais e à segurança de software. Essa etapa proporcionou a base para o desenvolvimento de um modelo ancorado na

técnica de *EventStorming* e no *Behavior-Driven Development (BDD)*, especificamente direcionado à análise da superfície de ataque.

Subsequentemente, o modelo desenvolvido foi submetido à aplicação e avaliação por meio de um relato de experiência, complementado por um estudo de caso. Este último integrou o *BDD* como estratégia para a validação dos requisitos de segurança, com especial atenção ao desempenho do sistema. A sinergia entre estas abordagens permitiu uma análise abrangente da eficácia da solução proposta, resultando em uma contribuição significativa para o avanço da engenharia de software orientada à segurança.

2

Referencial Teórico

Neste capítulo, serão abordados os conceitos essenciais que fundamentam o tema desta pesquisa, visando estabelecer a base teórica necessária para sua compreensão detalhada e precisa.

2.1 Ciclo de Desenvolvimento de Software

A inovação no desenvolvimento de software e na gestão de projetos ocupa uma posição central na indústria contemporânea. Esse processo é intrinsecamente delineado por fases sucessivas que compõem o ciclo de vida do sistema, estendendo-se do planejamento estratégico inicial à entrega e implementação final da solução. É nesse contexto que engenheiros e especialistas em software têm direcionado suas pesquisas para a integração de abordagens práticas, visando aprimorar cada uma dessas etapas. Nesta proposta, focaremos em três fases principais:

- Designer: Propomos a adoção do *EventStorming* para a análise da superfície de ataque, buscando identificar recursos suscetíveis a vulnerabilidades. Essa abordagem permite modelar a superfície de ataque, analisar as operações e comportamentos do sistema, e, assim, fornecer *insights* sobre resultados de medições relacionadas à segurança da informação, além de apontar possíveis recomendações (CHRISTOFOROU et al., 2017).

  Para essa análise, frequentemente se utiliza uma modelagem que incorpora o Domain-Driven Design (DDD), por possibilitar a identificação dos diversos componentes e interações dentro do sistema, o que favorece a identificação de áreas vulneráveis (BELLOVIN, 2016). A modelagem de sistemas voltada à análise de ataques superficiais visa desenvolver representações que permitam avaliar o funcionamento do sistema e seu uso pelos usuários, possibilitando a implementação de práticas de segurança que minimizem falhas. Contudo, o DDD não é amplamente adotado em contextos de microsserviços. Nesse cenário, o EventStorming, proposto por Brandolini, destaca-se como uma técnica colaborativa que

auxilia as equipes a compreender a lógica de negócio e os fluxos de eventos em um domínio específico (BRANDOLINI, 2013).

Essa abordagem visual envolve a criação de um modelo de eventos de domínio, tornando-se uma ferramenta valiosa para a análise da superfície de ataque. Durante sua aplicação, são identificados usuários, comandos e eventos disparados, bem como mapeados os fluxos entre componentes, o que permite a detecção de possíveis alvos de ataque.

- Implementação: Extrairemos as boas práticas sugeridas no artigo (Rezaei Nasab et al., 2023), aliadas à metodologia Behavior-Driven Development (BDD), para a construção de um novo estudo de caso.

- Teste: Utilizaremos o Behavior-Driven Development (BDD) (Rezaei Nasab et al., 2023), que permite realizar análises tanto durante quanto após o desenvolvimento, por meio da automação. O artigo (FUCCI et al., 2024) servirá como base para justificar os testes de segurança fundamentados na metodologia *OWASP*, a fim de verificar a existência de falhas.

A adoção de um modelo de ciclo de vida adequado é crucial para compreender as necessidades reais do software e planejar seu desenvolvimento de forma mais eficaz. A escolha do modelo determina a sequência de atividades a serem seguidas, garantindo a qualidade do software desde as etapas iniciais. Isso não só evita custos elevados com correções tardias, mas também assegura um melhor desempenho e disponibilidade para futuras melhorias ou correções.

Diversos modelos de ciclo de vida auxiliam no desenvolvimento de software e contribuem significativamente para a engenharia de requisitos. Em termos gerais, o ciclo de vida do software pode ser dividido em três fases básicas:

- Definição: Visa compreender a situação e identificar o problema, com o objetivo de propor uma solução adequada.

- Desenvolvimento: Envolve atividades relacionadas ao design, prototipagem, codificação, testes, entre outras consideradas essenciais. É importante ressaltar que essas ações devem estar alinhadas com o que foi definido nas fases iniciais.

- Operação: Consiste no suporte contínuo aos usuários, com foco na correção de possíveis falhas que possam surgir durante o uso do sistema.

  Encontrar a solução mais adequada para cada necessidade é fundamental para garantir o alinhamento entre o desenvolvimento de software e os requisitos dos usuários. Diante do crescente uso da tecnologia e da exposição de informações no ambiente virtual, as organizações precisam se preparar para se proteger e mitigar riscos potenciais. Em resposta a esse cenário, foram desenvolvidos métodos específicos para promover o desenvolvimento seguro de aplicações e sistemas em geral.

2.2 Ciclo de Desenvolvimento Seguro

A crescente sofisticação das ameaças cibernéticas exige uma abordagem proativa e estruturada para assegurar a resiliência e proteção dos sistemas. Nesse contexto, o Ciclo de Desenvolvimento Seguro (SSDLC) (UMEUGO, 2023), emerge não apenas como uma metodologia de desenvolvimento de software, mas como um paradigma que integra a segurança por meio da adoção de boas práticas em todas as fases do ciclo de vida do software.

Incorporar a segurança no desenvolvimento de software (HASSAN; DAS; HUSSAIN, 2023), desde as fases iniciais é crucial.Essa abordagem proativa permite a eliminação precoce de vulnerabilidades conhecidas, o que contribui significativamente para o aumento da segurança e a mitigação de riscos de ataques maliciosos. Consequentemente, torna-se essencial que os desenvolvedores apliquem, de forma contínua e sistemática, os conhecimentos mais atualizados concernentes a riscos, ameaças e vulnerabilidades.

Para atender a essa demanda, propomos a adoção de um conjunto de atividades estruturadas de segurança (SAEED et al., 2025), organizadas nas seguintes fases:

- Requisitos: Consiste na identificação e documentação dos requisitos de segurança da aplicação, assegurando que sejam considerados desde o início do projeto.
- Design: Nesta etapa, aplicam-se princípios de design seguro e realiza-se a modelagem de ameaças, com base em uma compreensão aprofundada das necessidades arquitetônicas do software.
- Desenvolvimento: Durante o desenvolvimento, é essencial que as equipes adotem padrões de codificação seguros e assegurem a conformidade com os requisitos de segurança previamente definidos.
- Teste: Incluem-se testes de segurança, tanto automatizados quanto manuais, com o objetivo de identificar vulnerabilidades e reforçar a proteção da aplicação.
- Operação: Após a implantação, as práticas de segurança devem continuar sendo observadas durante a manutenção do sistema. A aplicação deve ser constantemente atualizada para proteger-se contra novas vulnerabilidades e adaptar-se a cenários de ameaça emergentes.

Os princípios do Ciclo de Desenvolvimento de Software Seguro (SSDLC), são intrinsecamente aplicáveis à Arquitetura Baseada em Microsserviços, dada a natureza distribuída e independente de seus componentes. A fragmentação dos sistemas em microsserviços pode, inclusive, facilitar a aplicação de controles de segurança específicos para cada serviço, embora exija uma coordenação robusta para manter a segurança do ecossistema como um todo.

2.3 Arquitetura Baseada em Microserviços

A Arquitetura Baseada em Microsserviços é um estilo arquitetural que organiza uma aplicação como uma coleção de serviços coesos, pequenos e independentes, com baixo acoplamento. Cada microsserviço foca em uma funcionalidade de negócio específica, sendo desenvolvido, implantado e mantido de forma autônoma.

No desenvolvimento de software atual, essa arquitetura (Martínez Saucedo et al., 2025), se consolidou como um paradigma fundamental. Isso se deve às diversas práticas e tendências que visam maximizar seus benefícios e mitigar desafios. A abordagem não só oferece flexibilidade essencial para o gerenciamento, mas também otimiza a escalabilidade individual dos componentes do sistema.

2.3.1 Microserviços

A arquitetura de microsserviços (ROCHA; SOARES; RODRIGUEZ, 2023) ,representa um estilo de desenvolvimento de software que promove a criação de módulos autônomos, denominados microsserviços. Essa abordagem emergiu como uma solução intrinsecamente alinhada ao desenvolvimento em ambientes de computação em nuvem (RODRIGUEZ et al., 2023), impulsionada pela necessidade de sistemas mais escaláveis, resilientes e ágeis . Ao permitir que múltiplos serviços operem de forma independente, sem impactar o restante da aplicação, essa arquitetura está intrinsecamente relacionada ao Princípio da Responsabilidade Única. Este princípio fundamental estabelece que cada serviço deve ter apenas uma responsabilidade bem definida, garantindo a coesão e facilitando a manutenção e escalabilidade do sistema como um todo.

2.3.2 Princípio da Responsabilidade Única

Cada microsserviço deve ser projetado com uma única responsabilidade, comunicando-se exclusivamente com outros microsserviços para a execução de tarefas específicas. Essa abordagem assegura que cada serviço seja robusto, flexível e de fácil manutenção. Por exemplo, um microsserviço pode ser responsável pela autenticação de usuários, enquanto outro gerencia o processamento de pagamentos. A criação de um microsserviço que combine tanto a autenticação quanto o manuseio de pagamentos configura uma violação ao Princípio da Responsabilidade Única, comprometendo a modularidade do sistema e dificultando sua manutenção e escalabilidade. Qualquer alteração em uma funcionalidade pode afetar negativamente a outra, resultando em maior complexidade e risco de falhas. Assim, a adesão a esse princípio é fundamental para garantir a eficiência, a coesão e a integridade da arquitetura baseada em microsserviços. O Princípio da Responsabilidade Única (SRP), conforme apresentado por (HAOYU; HAILI, 2012), estabelece que um módulo ou componente deve ter apenas uma razão para ser modificado. Essa premissa é central na arquitetura de microsserviços, pois garante que cada serviço mantenha o

foco em uma funcionalidade específica, facilitando a identificação de falhas e a implementação de melhorias. Complementarmente, o desacoplamento emerge como um dos pilares dessa arquitetura, permitindo que cada serviço opere de forma autônoma, minimizando interdependências e, consequentemente, facilitando a evolução contínua do sistema.

2.3.3 Desacoplamento

O princípio do desacoplamento (MUDIMBA; PHIRI, 2024) , é fundamental na engenharia de software, pois enfatiza a importância de minimizar as dependências entre os componentes, permitindo que cada parte do sistema funcione de forma independente. Esse princípio está intimamente relacionado ao princípio da responsabilidade única, segundo o qual um módulo ou classe deve ter apenas uma razão para ser modificado ou seja, deve se responsabilizar por uma única funcionalidade. A aplicação combinada desses princípios promove uma arquitetura mais organizada e modular, na qual alterações em um componente impactam minimamente os demais. Isso não apenas simplifica a manutenção e evolução do software, como também favorece a testabilidade, a reutilização de código e o desenvolvimento de soluções mais robustas e adaptáveis.

2.4 Comunicação entre Microserviços

Ao tratar da arquitetura de microsserviços e dos seus modos de comunicação (WEERASINGHE; PERERA, 2022), é importante considerar duas abordagens distintas: a comunicação síncrona e a comunicação assíncrona. Cada uma apresenta vantagens e desvantagens, e a escolha entre elas deve ser orientada pelos requisitos específicos do sistema, tais como tolerância a falhas, tempo de resposta e necessidade de escalabilidade.

2.4.1 Comunicação Síncrona

O estilo de comunicação entre microsserviços baseado em protocolos síncronos (LIM, 2019), caracteriza-se pelo fato de um serviço realizar uma requisição diretamente a outro e aguardar sua resposta antes de prosseguir com o processamento. Trata-se de uma abordagem relativamente simples de implementar e apropriada em cenários nos quais os serviços dependem de respostas imediatas para dar continuidade às suas operações. No entanto, esse modelo pode resultar em um acoplamento mais forte entre os serviços, além de apresentar vulnerabilidades relacionadas à latência e à indisponibilidade, caso o serviço requisitado esteja fora do ar ou apresente instabilidades.

2.4.2 Comunicação Assíncrona

A comunicação de forma independente, na qual os serviços não requerem uma resposta imediata, é uma prática comum em arquiteturas distribuídas. Geralmente, essa comunicação ocorre por meio de filas de mensagens *message queues* ou intermediários de mensagens, o que contribui significativamente para a resiliência do sistema. Esse modelo permite que os serviços continuem a processar suas tarefas mesmo quando o serviço destinatário não está imediatamente disponível. A comunicação assíncrona (WOLFF, 2019), é amplamente adotada em cenários que exigem alta escalabilidade e nos quais a prontidão da resposta não constitui um requisito crítico.

2.5 Benefícios Arquiteturais

A arquitetura de microsserviços oferece uma série de benefícios substanciais (LAURETIS, 2019). Ela facilita a adoção de tecnologias heterogêneas, capacitando as equipes de desenvolvimento a selecionar as ferramentas mais adequadas para cada serviço específico. Adicionalmente, essa flexibilidade otimiza a eficiência das equipes, que podem operar de maneira mais autônoma e especializada. Consequentemente, a arquitetura de microsserviços viabiliza a entrega contínua de novas funcionalidades, acelerando o ciclo de inovação e a capacidade de resposta às demandas do mercado.

2.5.1 Escalabilidade Independente

A arquitetura de microsserviços (RAJ; SADAM, 2021) tem ganhado popularidade no desenvolvimento de software devido à sua capacidade de proporcionar flexibilidade e eficiência em sistemas complexos. Um de seus principais benefícios é a escalabilidade independente, que permite a cada microsserviço ser escalado de forma autônoma, conforme sua demanda específica. Diferentemente dos sistemas monolíticos, nos quais a aplicação é escalada integralmente, a arquitetura de microsserviços possibilita um ajuste granular dos recursos. Por exemplo, um microsserviço de autenticação pode exigir mais recursos durante picos de acesso, enquanto um de processamento de pagamentos pode ter demanda mais constante. Essa capacidade de escalar serviços de forma independente otimiza o uso de recursos e aumenta a resiliência do sistema, uma vez que falhas em um microsserviço não afetam diretamente os demais.

2.5.2 Implantação de Desenvolvimentos Independentes

A implantação de desenvolvimentos independentes (OMMERING, 2001) , refere-se à capacidade de publicar e atualizar diferentes partes de um sistema de software de forma autônoma, sem que uma implantação afete ou exija a implantação de outras partes. Essa abordagem é um pilar fundamental em arquiteturas modernas, especialmente na Arquitetura Baseada em Microsserviços. Ela complementa o princípio da resiliência, ao permitir que componentes do

sistema operem e sejam atualizados de maneira isolada. Isso cria as condições para que cada parte do sistema possa se adaptar a mudanças e superar desafios de forma independente, contribuindo para a estabilidade e a disponibilidade contínua da aplicação como um todo.

2.5.3 Resiliência

Yin et al. (YIN; DU, 2020) defendem que não existe uma definição clara e consensual sobre resiliência na área de software, o que gera incertezas para os desenvolvedores quanto ao nível de resiliência necessário e aos mecanismos a serem implementados. Para abordar essa questão, eles sugerem um modelo de medição de resiliência voltado especificamente para microsserviços. Esse modelo facilita a quantificação da resiliência em arquiteturas baseadas em microsserviços e permite a decomposição de metas de resiliência em comportamentos práticos, passíveis de serem executados pelos componentes do sistema.

A resiliência pode ser entendida como a capacidade de um sistema de manter suas funções essenciais mesmo na presença de falhas, adaptando-se a situações adversas e a desafios operacionais. Esse atributo é crucial para assegurar a continuidade e a eficácia de arquiteturas distribuídas, especialmente em ambientes complexos e dinâmicos. Nesse contexto, a utilização do processamento assíncrono assume papel importante, pois possibilita que tarefas e operações sejam executadas de forma não bloqueante, desvinculando-as de um fluxo sequencial rígido. Dessa forma, o sistema maximiza a utilização de seus recursos e reage rapidamente a eventos externos.

2.5.4 Antifragilidade

Taleb (TALEB, 2012) define um sistema antifrágil como aquele que se beneficia do caos, da incerteza e dos acontecimentos imprevistos, fortalecendo-se ao longo do tempo. Nas arquiteturas de microsserviços, a antifragilidade refere-se à habilidade do sistema não apenas de resistir a adversidades, mas de crescer e se fortalecer em resposta a elas.

A estrutura modular e independente dos microsserviços, caracterizada pela granularidade e distribuição, cria um ambiente propício à implementação de propriedades antifrágeis. Isso permite que o sistema absorva e aprenda com as falhas, adaptando-se e aprimorando-se continuamente, sem comprometer sua integridade ou desempenho geral. Assim, a antifragilidade em microsserviços não só protege o sistema contra falhas, mas também favorece seu crescimento e evolução constantes.

2.6 Segurança em Arquiteturas Distribuídas

Em arquiteturas distribuídas (JIA, 2025), a segurança constitui um desafio de complexidade notável, inerente à própria natureza desses sistemas. A proliferação de múltiplos

componentes interconectados e a extensão das redes amplificam substancialmente a superfície de ataque, elevando, assim, o número de potenciais pontos de vulnerabilidade.

Para a mitigação (HAINDL; KOCHBERGER; SVEGGEN, 2024) , eficaz desses riscos, a segurança deve ser concebida de maneira abrangente e multifacetada, abrangendo todas as camadas da arquitetura. Isso engloba desde a infraestrutura subjacente (incluindo servidores, redes e virtualização) até as aplicações e os dados que são processados e armazenados no ambiente distribuído . A negligência de qualquer uma dessas camadas pode resultar na criação de lacunas exploráveis, comprometendo a integridade e a confidencialidade do sistema em sua totalidade.

Ao adotar uma abordagem holística de segurança, que considera todos os aspectos da arquitetura distribuída, torna-se possível mitigar os riscos e garantir a proteção efetiva dos sistemas e dados. Essa perspectiva é crucial, pois permite uma identificação e tratamento mais abrangentes da superfície de ataque, que em ambientes distribuídos é intrinsecamente expandida pela multiplicidade de componentes interconectados.

2.6.1 Análise da Superfície de Ataque

A Análise da Superfície de Ataque é um conceito fundamental em segurança da informação, que envolve a avaliação sistemática de possíveis entradas ou pontos de acesso que um agente mal-intencionado pode explorar em um sistema (CHEN, 2019). Seu objetivo primordial é identificar e mapear todos os caminhos pelos quais dados ou componentes podem ser acessados ou exfiltrados, visando comprometer a confidencialidade, integridade ou disponibilidade dos ativos.

Essa análise foca na identificação dos limites de confiança e dos pontos de interação (entradas/saídas de dados) que podem se transformar em vetores de ataque (JAYALATH et al., 2024). Quando relacionada a um Diagrama de Fluxo de Dados (DFD), ela complementa a representação visual ao ilustrar o movimento dos dados entre os processos. Assim, é possível discernir onde validações são cruciais, quais dados são sensíveis, e onde um invasor poderia inserir ou extrair informações de forma não autorizada, transformando cada interface em um potencial ponto de vulnerabilidade a ser mitigado.

2.6.2 Diagrama de Fluxo de Dados (DFD)

DeMarco (DEMARCO, 2001) defendia o Diagrama de Fluxo de Dados (DFD) como uma ferramenta central na análise estruturada de sistemas. Contudo, sua utilidade transcende essa esfera inicial, atuando como uma ferramenta visual versátil para a compreensão e documentação de sistemas, análise de requisitos, identificação de ineficiências, otimização de processos e análise de segurança, entre outras aplicações.

Embora o Diagrama de Fluxo de Dados tenha suas raízes firmemente estabelecidas na

análise estruturada, sua evolução, impulsionada pelas contribuições de diversos especialistas em sua formalização e popularização, o transformou em uma ferramenta multifuncional. Ele é empregado em variadas fases do ciclo de vida do desenvolvimento de software e na análise de negócios, particularmente em contextos que demandam clareza sobre o fluxo de informações. Ao permitir a compreensão do fluxo de dados em um sistema, o *DFD* facilita a identificação de problemas, a definição precisa de requisitos e o planejamento eficiente de testes, culminando na garantia da qualidade do software, que se torna mais robusto, confiável e de fácil manutenção.

2.7 Modelagem e Qualidade

A modelagem de processo consiste na representação de abstrações definidas em alinhamento com os objetivos e interesses das partes envolvidas nos processos (BHUTA; BOEHM; MEYERS, 2005). Por sua vez, a qualidade refere-se ao grau em que um produto, serviço ou processo atende às expectativas e necessidades. Embora conceitos distintos, eles se complementam, formando uma poderosa sinergia na gestão organizacional. Enquanto a modelagem de processo (ROCHA; MISRA; SOARES, 2023), descreve "como" as atividades são executadas, a qualidade se concentra no "quão bem" essas atividades são realizadas e nos resultados que esses conceitos capacitam as organizações a operar de forma mais eficiente e previsível, entregando valor consistente aos seus *stakeholders*.

Nesse contexto, a união da modelagem de processo com os princípios de qualidade encontra uma aplicação sinérgica no *EventStorming*. Essa técnica, ao facilitar a representação dos fluxos operacionais, permite a identificação precoce de requisitos de segurança e pontos de vulnerabilidade. Dessa forma, o *EventStorming* atua como um catalisador para a criação de sistemas mais robustos e seguros, alinhando a estrutura dos processos com os objetivos de desempenho e confiabilidade.

2.7.1 EventStorming

O *EventStorming*, uma metodologia de *workshops* que emprega *post-its*, distingue-se por sua natureza adaptativa. Tal característica fomenta uma colaboração interdisciplinar sofisticada entre *stakeholders* com *expertises* diversas, o que facilita a exploração eficiente de domínios de negócio complexos. Consequentemente, essa abordagem viabiliza a compreensão do sistema por meio de questionamentos que geram resultados significativos (BRANDOLINI, 2013).

A interatividade da técnica é evidenciada pela função específica atribuída a cada cor de post-it, permitindo identificar potenciais pontos de interação do software. Os elementos principais incluem:

- Eventos (representados por *post-its*[1] laranja, com sujeito e verbo no pretérito), visam a

[1] **Post-it** é uma marca registrada. O termo genérico correspondente é **nota autoadesiva**.

clareza na comunicação dos acontecimentos do sistema e dos dados compartilhados, com ênfase na confiabilidade e segurança da informação.

- Dúvidas (rosas), que frequentemente indicam pontos críticos relacionados ao acesso, integridade de dados e conformidade, sinalizando requisitos de segurança.
- Comandos (azuis , com verbo no infinitivo), gerados a partir dos eventos e responsáveis por sua ativação.
- Atores (amarelo-escuro), simbolizando os usuários que interagem com o sistema por meio dos comandos, o que é crucial para a modularização do sistema.
- Políticas (lilás), que representam condições ou conjuntos de condições iniciadas por eventos para a execução de comandos específicos, e cuja violação pode induzir vulnerabilidades ou fragilidades na segurança.
- Agregadores (amarelo-claro), que reúnem comandos e eventos, conectando seus contextos e auxiliando na definição de *bounded contexts*, otimizando esforços de refatoração e reduzindo a carga cognitiva das equipes.

A relevância do *EventStorming* no processo de desenvolvimento de software é significativamente ampliada ao considerar padrões de qualidade como a *ISO/IEC 25010:2023*. Esta norma internacional, que define um modelo abrangente para a qualidade de produto de software, oferece um arcabouço para especificar e avaliar requisitos de qualidade. A estrutura do *EventStorming*, portanto, provê um mecanismo proativo para a elicitação de requisitos de segurança, pois a identificação de eventos, comandos e atores permite mapear a superfície de ataque potencial de um sistema. As dúvidas e políticas, como mencionado, são ferramentas que expõem implicitamente os pontos onde a proteção da informação (abrangendo confidencialidade, integridade e disponibilidade) e a gestão de riscos se tornam essenciais. Essa abordagem facilita a identificação e o refinamento dessas características de segurança de maneira colaborativa e visual, alinhando-se diretamente às diretrizes da (Technical Committee: ISO/IEC JTC 1/SC 7, 2023).

2.7.2 ISO/IEC 25010:2023

A avaliação da segurança em sistemas emprega técnicas e ferramentas específicas para mensurar a conformidade com os critérios da *ISO/IEC 25010:2023*. A adoção dessas práticas aprimora o processo de desenvolvimento, eleva a satisfação dos usuários e aumenta a competitividade no setor de software.

Nesse contexto, um processo de software é concebido como um conjunto de atividades organizadas para atingir um objetivo definido, cuja capacidade reflete a habilidade de alcançar resultados consistentes e a maturidade denota o grau de evolução e institucionalização das

práticas. Tais atributos são cruciais para a melhoria contínua da qualidade do software (STEINDL et al., 2024) e devem espelhar as necessidades das partes interessadas. Os modelos de maturidade apoiam a evolução dos processos com diretrizes para sua implementação e aprimoramento. Assim, a maturidade em segurança de software refere-se à capacidade de um sistema proteger informações e dados contra vulnerabilidades, bem como gerenciar os impactos de possíveis ataques.

Com a evolução e o consequente aumento da complexidade dos sistemas, a garantia da confiabilidade torna-se um desafio progressivo. Essa intrínseca complexidade pode resultar em erros, atrasos na entrega e custos que frequentemente excedem as estimativas iniciais. Nesse contexto, a arquitetura de software emerge como um pilar fundamental, provendo uma visão de alto nível que serve de alicerce para o desenvolvimento de diversas técnicas de modelagem, design e linguagens de programação. Contudo, o crescente uso de software, em conjunto com a expansão de seu tamanho e complexidade, impõe desafios substanciais à análise e à realização de testes aprofundados, impactando diretamente os custos de manutenção.

2.7.3 Behavior-Driven Development

O *Behavior-Driven Development (BDD)*, desenvolvido por Dan North em 2003 (NORTH, 2006), constitui um *framework* destinado a simplificar os desafios inerentes ao ciclo de vida de um sistema, promovendo a eficácia e a eficiência na comunicação e na entrega de requisitos. Em 2009, a introdução da técnica dos "3 amigos" buscou integrar perspectivas de negócio, desenvolvimento e testes, solidificando a proposta original do *BDD*.

Como prática ágil, o *BDD* (RAHARJANA; HARRIS; JUSTITIA, 2020) foca na entrega do comportamento essencial para a composição de sistemas. A consequente economia de tempo, advinda da redução do retrabalho, permite que as equipes direcionem seus esforços para outras atividades cruciais do desenvolvimento de software. Tal abordagem alinha-se intrinsecamente à Cultura Ágil, que emergiu no cenário empresarial após as transformações digitais, fomentando agilidade, qualidade e adaptabilidade por meio de um ambiente de trabalho dinâmico, participativo e cooperativo.

O processo de desenvolvimento no *BDD* fundamenta-se na elaboração de cenários de teste utilizando a linguagem *Gherkin (Given/When/Then)*. Essa linguagem permite descrever o percurso do usuário em uma aplicação e o comportamento esperado do sistema. A documentação gerada é dinâmica, assegurando a agilidade necessária para atualizações contínuas do processo. Em virtude desses atributos, o *BDD* proporciona economia de tempo e recursos, visto que minimiza o retrabalho decorrente de entregas de baixa qualidade, além de oferecer uma validação aprimorada (BINAMUNGU; EMBURY; KONSTANTINOU, 2018; MOE, 2019) e um entendimento mais profundo do código (GUERRA-GARCIA et al., 2023).

A eficácia do *BDD* pode ser significativamente ampliada com a integração de *Large*

Language Models (LLMs), como o *ChatGPT* ou *Gemini*. *LLMs* podem atuar como ferramentas poderosas para auxiliar na geração de cenários *Gherkin*, a partir de requisitos de alto nível ou conversas com stakeholders. Eles são capazes de processar descrições em linguagem natural e sugerir múltiplos cenários "Dado-Quando-Então" , garantindo maior cobertura e consistência. Além disso, *LLMs* podem ajudar na validação e refatoração de cenários existentes, identificando ambiguidades ou inconsistências, e até mesmo na geração de código de teste correspondente aos cenários *Gherkin*. Essa colaboração entre *BDD* e *LLMs* promete otimizar ainda mais o processo de desenvolvimento, acelerando a criação de especificações claras e a automação de testes, o que, por sua vez, contribui para a entrega de software de maior qualidade e com menor custo.

2.7.4 Large Language Models (LLMs)

Modelos de Linguagem de Grande Escala (LLMs), como o ChatGPT, desenvolvido pela OpenAI (OpenAI, 2024), e o *Gemini*, criado pelo *Google DeepMind* (DEEPMIND, 2024), têm revolucionado a interação homem-máquina. Sua capacidade notável de simular conversas e compreender nuances contextuais permite uma comunicação mais fluida, natural e intuitiva.

O *ChatGPT*, um exemplo proeminente dessa inovação, fundamenta-se na arquitetura *Transformer* (RADFORD et al., 2019), a qual revolucionou o campo do processamento de linguagem natural. Essa estrutura capacita o modelo a compreender contextos complexos e a gerar respostas detalhadas e coerentes. Alimentado pela avançada arquitetura *GPT-4* e treinado em um vasto conjunto de textos, ele se destaca por sua habilidade em lidar com uma vasta gama de tópicos e manter diálogos interativos. O modelo aborda questões de forma relevante e contextualizada, adaptando-se a diversos tipos de interação desde assistentes virtuais até ferramentas de geração de conteúdo o que o posiciona como um recurso disruptivo com amplas aplicações em áreas como educação, atendimento ao cliente e desenvolvimento de conteúdo.

3

Trabalhos Relacionados

De acordo com Hernandez et al. (2010), a ascensão da inovação no desenvolvimento de software e no gerenciamento de projetos tem sido um foco constante de pesquisa para empresas de software e terceirizadas, com o objetivo de aprimorar as fases de desenvolvimento, análise, design, teste e implantação. Fornecer ferramentas que simplifiquem essas etapas e permitam a contribuição de qualquer pessoa de forma prática, utilizando um método focado no comportamento (BRANDOLINI, 2013), torna o teste mais fácil de manter, por ter um objetivo mais claro. Isso proporciona ao *BDD* um entendimento mútuo, no qual o analista requisita o que precisa ser feito, o desenvolvedor sugere soluções e o *Quality Assurance (QA)* identifica cenários e situações ainda não contempladas.

O estudo empírico Rezaei Nasab et al. (2023) revela que, apesar dos inúmeros benefícios dos sistemas de microsserviços, a segurança continua sendo uma questão crítica. Essa análise resultou em um catálogo de 28 práticas de segurança para microsserviços, que pode servir como um recurso valioso para profissionais da área. A disponibilização dessas diretrizes depende de uma mudança comportamental, pois constituem apenas parte de um processo cuja adoção de estratégias visa superar barreiras e fomentar pesquisas em áreas ainda pouco exploradas.

Zhang (2023) discute a solução *IAST*, propondo um estudo comportamental fundamentado em aprendizado de máquina *(Machine Learning)*. Esse aprendizado se desenvolve por meio da localização e exploração de vulnerabilidades em tempo real, com base em microsserviços voltados à garantia da segurança. Assim, a *IAST* visa instituir compatibilidade com a segurança por meio de políticas orientadas à detecção de explorações de vulnerabilidades, falsos positivos e anomalias. A proposta é construir confiança, aumentando a acurácia e viabilizando o monitoramento do status de segurança em tempo real.

Reselman (2022a), Reselman (2022b), Reselman (2022c) apresenta cinco princípios de design para microsserviços: responsabilidade única (cada microsserviço possui apenas uma responsabilidade), limites discretos (o microsserviço é encapsulado e separado de seu ambiente),

transportabilidade (podem ser transferidos para diferentes ambientes com pouco esforço), domínio próprio (carregam seus próprios dados) e efemeridade (são criados e destruídos conforme a necessidade da aplicação).

Um estudo sistemático conduzido por Rezapour et al. (2021) apresenta os princípios fundamentais do *fog computing*, delineando os critérios necessários para assegurar a proteção desse ambiente computacional. O *fog computing*, enquanto tecnologia emergente, destaca-se por processar e analisar dados com eficiência e baixa latência, reduzindo a dependência da nuvem. Essa abordagem descentralizada mitiga congestionamentos na rede e melhora o tempo de resposta dos sistemas distribuídos.

O estudo de Roman, Lopez e Mambo (2018) fornece uma análise holística da segurança dos paradigmas de borda, considerando ameaças e desafios que afetam paradigmas como computação em névoa, computação de borda móvel e computação em nuvem móvel.

Levcovitz, Terra e Valente (2016) propõe uma técnica para identificar e definir serviços em ambientes monolíticos empresariais, partindo do princípio de que grandes sistemas são estruturados em subsistemas menores. Aplicativos monolíticos possuem três partes principais: interface do usuário, aplicação no lado do servidor e banco de dados. Com a divisão em subsistemas, cada um com responsabilidades específicas, é possível identificar os "candidatos"a microsserviços. A arquitetura monolítica, embora eficaz inicialmente, torna-se desafiadora com o crescimento e a escalabilidade, por ser executada como um único processo em toda a base de código. Microsserviços, ao contrário, são caracterizados pelo desacoplamento e descentralização, permitindo o processamento independente.

Herrmann (2001) propõe uma abordagem de modelagem e análise de segurança para sistemas orientados a objetos por meio de análise de fluxo de informações, utilizando técnicas de modelagem baseadas em *UML* para avaliar a superfície de ataque.

Jürjens, Schreck e Bartmann (2008) propuseram o método *UMLSec*, que combina a modelagem de requisitos de segurança com a análise de aplicações.

Georg et al. (2010) apresentaram a metodologia *Aspect-Oriented Risk-Driven Development (AORDD)* para análise formal da segurança de aplicações, destacando como vantagem o equilíbrio entre requisitos de segurança e fatores como tempo de mercado e orçamento.

Bradbury et al. (2020) introduz uma arquitetura de referência *(RA)* para modelar aplicações aeroespaciais e identificar a superfície de ataque do sistema. A análise é feita por meio de árvores de ataque que especificam o caminho necessário para que um agente de ameaça atinja uma meta.

Rajbhoj et al. (2024) realizaram um estudo de caso utilizando o *ChatGPT* na etapa de elicitação de requisitos, com o objetivo de otimizar o tempo e permitir maior foco em outras fases do desenvolvimento. Os autores apontam a *IA* generativa como uma área promissora para futuras pesquisas. Nesse contexto, o experimento deste artigo busca contribuir com essa linha de investigação ao avaliar a eficácia do *ChatGPT* na automação de testes, ampliando a compreensão

sobre seu impacto na qualidade do desenvolvimento de software.

Santos et al. (2024b) conduziram um estudo de caso para avaliar a efetividade do *Behavior-Driven Development (BDD)* na captura de requisitos não funcionais, com ênfase em desempenho. Os resultados indicaram que o *BDD* é eficaz na elicitação desses requisitos e no alcance de metas. Com isso em mente, o experimento atual visa ampliar essa investigação ao analisar a elicitação de requisitos não funcionais conforme a norma *ISO/IEC 25010:2023*, utilizando Modelos de Linguagem de Grande Escala *(LLMs)*. Caso os resultados sejam positivos, essa abordagem pode otimizar o tempo no processo de desenvolvimento de software.

Olsson, Sentilles e Papatheocharous (2022) realizaram uma Revisão Sistemática da Literatura sobre pesquisas empíricas relacionadas à elicitação de requisitos não funcionais. Os autores destacaram que a norma *ISO/IEC 25010:2023* desempenha um papel fundamental na orientação da qualidade de software, embora haja necessidade de soluções mais eficazes para requisitos não funcionais. Neste contexto, o experimento apresentado neste artigo visa contribuir por meio do uso do *Behavior-Driven Development (BDD)* na identificação e automação de testes de requisitos não funcionais, empregando um *Large Language Model (LLM)* para aprimorar a qualidade do software e otimizar o tempo de desenvolvimento.

3.1 Analise Comparativa

Nesta seção, realizamos uma análise comparativa detalhada dos trabalhos científicos previamente selecionados no estado da arte. O objetivo principal é contextualizar a presente dissertação no cenário acadêmico e evidenciar suas contribuições e diferenciais. Para tal, os estudos foram caracterizados e confrontados em termos de seus objetivos, metodologias, principais resultados e limitações, conforme sumarizado no Quadro 1. Esta abordagem permite identificar lacunas na literatura e justificar a relevância e originalidade da pesquisa aqui desenvolvida.

Figura 1 – Quadro comparativo entre Trabalhos Relacionados e a Dissertação

Característica	**Rezaeinasab et al. (Rezaei Nasab et al., 2023)**	**Alkhuzaie et al. (HERNANDEZ et al., 2010)**	**Zhang et al. (ZHANG, 2023)**	**Reselman et al. (RESELMAN, 2022a; RESELMAN, 2022b; RESELMAN, 2022c)**	**Esta pesquisa**
Objetivo Principal	Explorar pesquisa e práticas da indústria em segurança de microsserviços, visando catálogo abrangente.	Propor metodologia para detecção e reconfiguração de ambientes de microsserviços, visando redução de vulnerabilidades.	Discute desafios de segurança e propõe IAST/ML para detecção acurada de vulnerabilidades.	Apresentar cinco princípios fundamentais de design para arquiteturas de microsserviços.	Propor modelo abrangente para análise de superfície de ataque e requisitos não funcionais em microsserviços.
Metodologia	Estudo empírico via análise de literatura, artigos e surveys/estudos de caso para coletar práticas de segurança.	Propõe metodologia de identificação, avaliação e mitigação, fundamentada em padrões de segurança (ex: Common Criteria, OWASP, NIST).	Pesquisa *survey* com revisão crítica da literatura sobre desafios e soluções de segurança em microsserviços, resultando em proposta.	Descreve princípios de design baseados em melhores práticas de arquitetura de software e experiência prática.	*Design Science Research (DSR)* em três fases: aquisição, construção e avaliação.
Principais Resultados/Contribuições	Criação de catálogo de 28 práticas de segurança, destacando criticidade na cultura de segurança em microsserviços.	Propõe metodologia para detecção e reconfiguração de ambientes de microsserviços, integrando padrões de segurança para diminuir vulnerabilidades.	Solução *IAST* com ML para detecção e exploração em tempo real, maior acurácia.	Cinco princípios essenciais para compreensão arquitetônica.	Melhorias na identificação/avaliação de vulnerabilidades e aspectos de segurança e qualidade.
Limitações	Fontes focadas na academia, sem validação empírica e resistência cultural à mudança.	Foco conceitual, difícil implementação, validação empírica limitada.	Validação de *ML*, dificuldades técnicas em tempo real.	Base em princípios teóricos, sem foco prático, pode desatualizar.	Limitações do *EventStorming* na cobertura de segurança, dependência de especialistas.
Diferenciais em Relação à Dissertação	Uso prático de *EventStorming*/BDD com *LLMs* na elicitação, validação e análise de superfície de ataque.	Foco na integração de segurança ao desenvolvimento (*BDD*, *EventStorming*, *LLMs*), complementando infraestrutura.	Abordagem proativa usando *EventStorming*, *BDD* e *LLMs* para criação de cenários e prevenção.	Apoio nos princípios de Reselman para garantir segurança, com foco em aplicação prática na dissertação.	

4

Design Science Research (DSR)

Esta dissertação adota a abordagem Design Science Research (DSR) como método principal de investigação, conduzido em três etapas complementares: aquisição de conhecimento, construção do modelo e avaliação.

O Mapeamento Sistemático de Literatura (MSL) desempenhou um papel vital, fornecendo o conhecimento fundamental para as fases subsequentes da metodologia *DSR*. As percepções obtidas na *MSL* não apenas informaram, mas também substanciaram a originalidade e a necessidade do modelo proposto: uma abordagem metodológica inovadora (HEVNER; CHATTERJEE, 2010) centrada na técnica de *EventStorming* e no *BDD*, projetada especificamente para identificar e avaliar a superfície de ataque em arquiteturas de software.

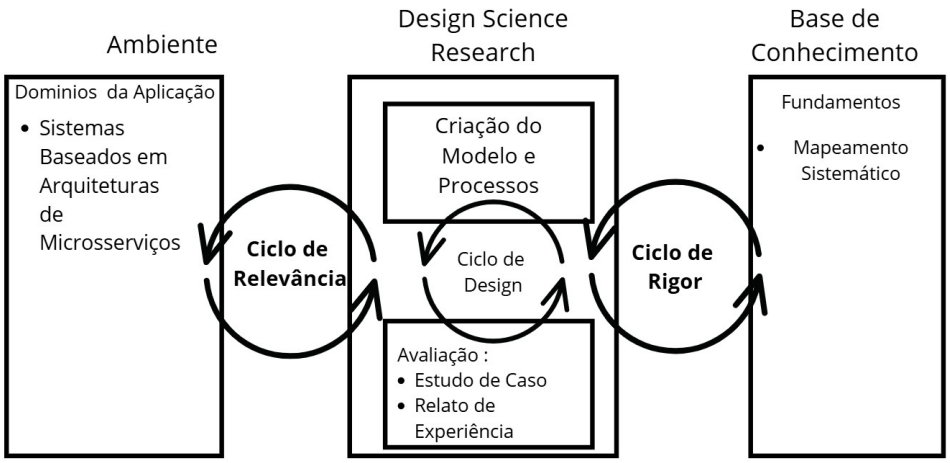
Essa abordagem alinha-se perfeitamente ao paradigma de resolução de problemas da DSR, conforme articulado por Brocke, Hevner e Maedche (BROCKE; HEVNER; MAEDCHE, 2020). Nesse contexto, essa metodologia transcende a mera concepção de soluções práticas para desafios concretos; ela contribui significativamente para o avanço do conhecimento científico ao gerar teorias de design (AKEN; ROMME, 2012). Tais teorias explicam os efeitos do modelo desenvolvido nos ambientes onde são aplicados, evidenciando suas contribuições e limitações intrínsecas. Desse modo, a *MSL* funcionou como a base empírica inicial, guiando o design de um modelo destinado a preencher as lacunas identificadas, o qual é então avaliado criticamente segundo seus princípios para gerar novos conhecimentos científicos.

Com base no modelo desenvolvido, realizamos um estudo prático por meio de um relato de experiência, observando a aplicação da abordagem em um cenário real e analisando seus benefícios e limitações. Para complementar a avaliação, conduzimos também um estudo de caso, integrando a técnica de *Behavior-Driven Development (BDD)* ao processo. Essa integração possibilitou a verificação sistemática de requisitos não funcionais que impactam a segurança, com ênfase no desempenho dos sistemas.

A combinação de métodos empíricos e construtivos proporcionou uma análise aprofun-

dada da eficácia da proposta (HEVNER; CHATTERJEE, 2010), contribuindo para o avanço do conhecimento em engenharia de software segura e orientada a requisitos.

Figura 2 – Design Science Research



Fonte: Tacyanne Pimentel (2025).

4.1 Mapeamento Sistemático de literatura

O Mapeamento Sistemático de Literatura *(MSL)* foi inicialmente proposto pelo médico britânico Archie Cochrane (1972), reconhecido como o pai da medicina baseada em evidências (SHAH; CHUNG, 2009). Cochrane desempenhou um papel fundamental na consolidação da ideia de que decisões clínicas e científicas devem ser fundamentadas em evidências obtidas por meio de análises rigorosas e sistemáticas da literatura existente.

Posteriormente, as diretrizes da *MSL* foram adaptadas ao contexto da engenharia de software por Barbara Kitchenham, com o objetivo de refletir as particularidades e os desafios dessa área de pesquisa (KITCHENHAM; CHARTERS, 2007).

A *MSL* configura-se como um método de investigação estruturado, cujo propósito é identificar, avaliar e interpretar criticamente todas as evidências científicas disponíveis e relevantes relacionadas a uma questão de pesquisa, área temática ou fenômeno específico de interesse. Essa abordagem proporciona maior confiabilidade e validade às conclusões obtidas, uma vez que segue protocolos metodológicos bem definidos, sistemáticos e reprodutíveis.

4.1.1 Critérios PICO

A metodologia *PICO* é amplamente utilizada na formulação de perguntas de pesquisa e na condução de buscas por evidências científicas, especialmente em áreas como a engenharia de software, por oferecer uma estrutura sistemática e clara que favorece investigações mais objetivas e rigorosas. Composta pelos elementos população (P), intervenção (I), comparação (C) e resultados (O), essa abordagem auxilia na delimitação do escopo da pesquisa, no refinamento dos termos de busca e na análise crítica das evidências coletadas (KITCHENHAM; CHARTERS, 2007).

A Tabela 1 apresenta a descrição dos componentes da estratégia PICO, destacando suas aplicações no contexto da engenharia de software e sua contribuição para a elaboração de recomendações fundamentadas em evidências.

Tabela 1 – Critérios PICO

Acrônimo	Critérios PICO	Descrição
P	População	Sistemas com arquitetura de microsserviços.
I	Intervenção	Análise da superfície de ataque.
C	Comparação	Não aplicável.
O	Resultados	Metodologia para análise e mitigação de superfícies de ataque.

4.1.2 Palavras-chave

A Tabela 2 apresenta as palavras-chave selecionadas para a formulação da sequência de busca empregada no Mapeamento Sistemático de Literatura descrita na Seção 4.1. Esses termos foram definidos com base em sua relevância para os objetivos e as questões da pesquisa, possibilitando a construção de uma estratégia de busca precisa e eficaz. Essa abordagem favoreceu a identificação de evidências científicas pertinentes, assegurando a seleção de estudos alinhados aos principais conceitos do trabalho e contribuindo para a confiabilidade e a abrangência dos resultados obtidos.

Tabela 2 – Palavras-chave utilizadas na *string* de busca

Palavra-chave	**Sinônimos em inglês**
attack surface	attack area, attack frontier, attack profile, attack vector, exposed surface, potential attack targets, surface susceptible to attacks, vulnerable entry/exit points, vulnerable surface
methodology	approach, method, methodical, methodologies, methods, procedure, process, technique
microservice	microservices

4.1.3 Questões de pesquisa

A análise da superfície de ataque em sistemas baseados em microsserviços configura-se como uma metodologia sistemática voltada à identificação de vulnerabilidades em componentes distribuídos, *APIs* e canais de comunicação. Ao considerar características específicas desses sistemas, como escalabilidade e comunicação descentralizada, essa abordagem contribui para o fortalecimento da segurança e da resiliência arquitetural. Nesse contexto, cinco questões de pesquisa foram definidas e estão organizadas na Tabela 9, com o propósito de orientar a investigação dos principais fatores relacionados à mitigação de riscos em ambientes de microsserviços.

4.1.4 Termos de pesquisa

Para ampliar e refinar os resultados, as palavras-chave foram combinadas por meio de operadores *booleanos* (*AND*, *OR* e *NOT*), permitindo a identificação de estudos relacionados à segurança, vulnerabilidades e estratégias de mitigação em ambientes de microsserviços. As *strings* de busca utilizadas estão apresentadas na Tabela 4, com exceção da base Science@Direct, que exigiu uma versão reduzida devido à limitação no número de caracteres, conforme indicado na Tabela 5.

Tabela 3 – Questões de Pesquisa

Questão	Descrição	Justificativa
Q1	Em que contexto os microsserviços vêm sendo aplicados?	Entender o contexto desses usos permite adaptar a análise de superfície de ataque às necessidades de segurança e às vulnerabilidades específicas de cada setor.
Q2	Quais as metodologias utilizadas?	Identificar as metodologias é essencial para determinar as abordagens mais eficazes.
Q3	Quais as condições para se utilizar superfícies de ataque?	Identificar as condições que justificam seu uso auxilia em uma aplicação mais eficaz e direcionada.
Q4	Quais os fatores condicionantes para se utilizar superfícies de ataque?	Entender os elementos técnicos e contextuais que influenciam a escolha e a eficácia da análise.
Q5	Quais os pontos positivos e os negativos?	Avaliar os benefícios e limitações permite uma visão balanceada, essencial para definir se essa metodologia é a mais indicada e onde ela pode ser melhorada.

Tabela 4 – *String* de busca utilizada nas bases de dados

String de Busca
(("microservice"OR "microservices") AND ("attack surface"OR "attack area"OR "attack frontier"OR "attack profile"OR "attack vector"OR "exposed surface"OR "potential attack targets"OR "surface susceptible to attacks"OR "vulnerable entry/exit points"OR "vulnerable surface") AND ("methodology"OR "approach"OR "method"OR "methodical"OR "methodologies"OR "methods"OR "procedure"OR "process"OR "technique"))

Tabela 5 – *String* reduzida utilizada na base Science@Direct

String de Busca
("microservice"OR "microservices") AND ("attack surface"OR "attack area"OR "attack frontier")

4.1.5 Bases utilizadas na pesquisa

A Tabela 6 apresenta a quantidade inicial de artigos recuperados nas bases de dados selecionadas para o Mapeamento Sistemático de Literatura. Ao todo, foram identificados 1.281

artigos relacionados aos temas de segurança, vulnerabilidades e metodologias em microsserviços.

Tabela 6 – Número de Artigos

Base	**Quantidade**
ACM Digital Library	206
IEEE Xplore	38
Web of Science	8
Scopus	102
Springer Link	845
Total	**1.281**

4.1.6 Seleção de pesquisa

Após a definição da questão de pesquisa, foram adotadas estratégias de busca para assegurar a recuperação sistemática de evidências relevantes. A seleção das fontes seguiu critérios específicos, apresentados na Tabela 7, enquanto a Tabela 6 lista as bases utilizadas. No total, 1.281 estudos publicados entre 2015 e 2024 compuseram o conjunto inicial de análise. A Figura 3 apresenta a distribuição desses estudos, destacando o *SpringerLink* e a ACM Digital *Library* como as principais fontes de dados.

Tabela 7 – Critérios de seleção

Código	Descrição
DSC01	Ser uma base de dados científica relevante e completa em Análise de Superfícies de Ataque em Sistemas com Arquitetura de Microsserviços.
DSC02	Disponibilidade de acesso ao conteúdo.

4.1.7 Estratégia de busca

A estratégia de busca foi organizada nas seguintes etapas:

1. Definição dos critérios de busca, extração dos termos e execução da pesquisa: inicialmente, 1.281 estudos foram incluídos. As palavras-chave e as combinações de termos foram cuidadosamente selecionadas para maximizar a relevância dos resultados nas bases de dados consultadas;

2. Remoção de duplicatas: após a eliminação dos estudos repetidos, restaram 1.157 trabalhos para análise subsequente;

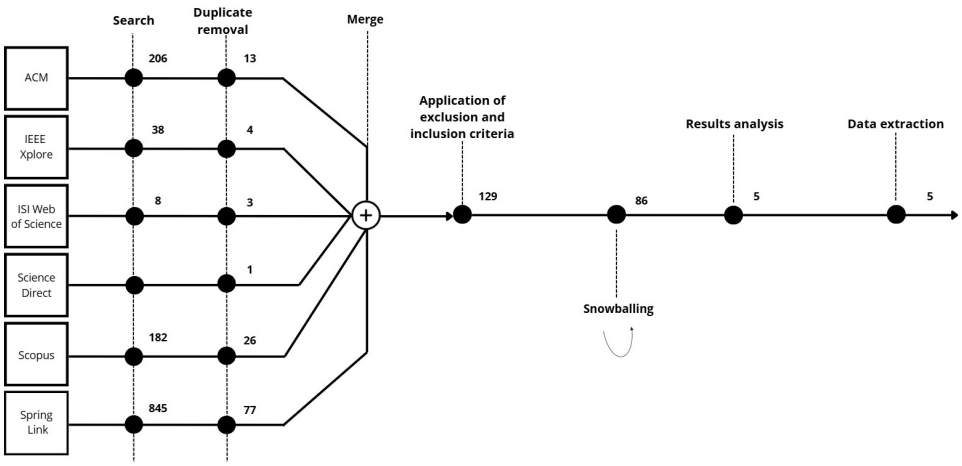
3. Primeira análise qualitativa: com base nos critérios de inclusão e exclusão, foram examinados títulos, resumos e palavras-chave, resultando na retenção de 277 estudos;

4. Realização da técnica de *snowball* reversa, extraindo as referências relevantes de cada estudo selecionado;

5. Segunda análise qualitativa: nesta etapa, os textos completos foram analisados em profundidade, culminando na seleção de 129 estudos considerados pertinentes. A lista completa desses estudos encontra-se na Tabela 13, disponível no Apêndice;

6. Extração de dados: Por fim, foram coletados os dados necessários para responder às questões de pesquisa. Dos 129 estudos inicialmente selecionados, 86 abordaram diretamente os temas investigados, porém apenas 8 respondem às questões de pesquisa 3;

Figura 3 – Gráfico de prisma com extração de dados



4.1.8 Critério

Critérios específicos foram estabelecidos para assegurar a seleção adequada das referências bibliográficas identificadas durante a busca, incluindo definições claras de critérios de inclusão e exclusão. Garantir a relevância e a qualidade das fontes é fundamental para a robustez do estudo. Ademais, não foi imposta nenhuma restrição temporal ao período de busca, o que possibilitou uma análise abrangente e atemporal dos dados disponíveis. Dessa forma, a Tabela 8 apresenta um detalhamento dos critérios de inclusão e exclusão adotados, evidenciando a metodologia rigorosa aplicada na seleção das referências.

Tabela 8 – Critérios de Inclusão e Exclusão

Critérios de Inclusão	**Critérios de Exclusão**
• Artigos que abordam fatores relacionados a microsserviços.	• Artigos duplicados. • Artigos que não estão escritos em inglês. • Artigos que não abordam, pelo menos parcialmente, algum dos requisitos estabelecidos. • Artigos que não tratam da arquitetura de microsserviços.

4.2 Modelo para analise de superficie de ataque

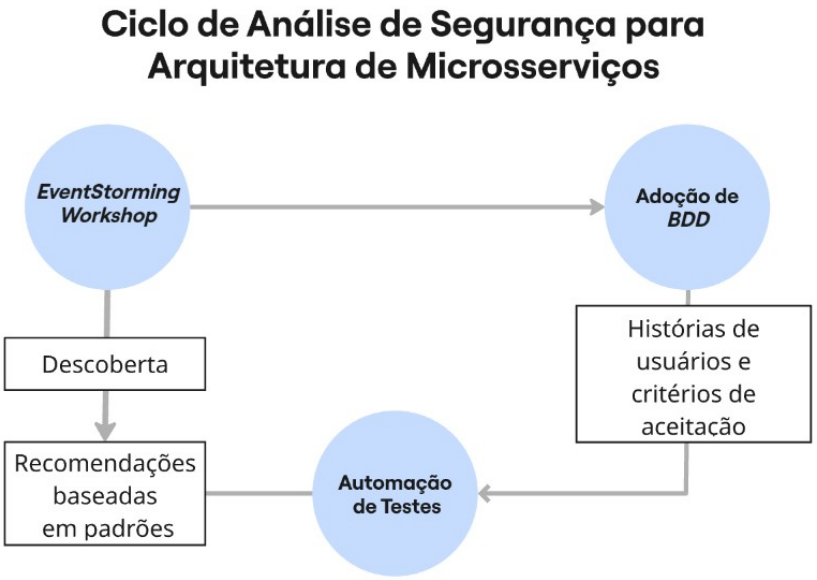
Com base nos dados coletados em um mapeamento sistemático, propõe-se um modelo de análise de segurança para arquiteturas de microsserviços. Esse ciclo, ilustrado na Figura 4, integra o *EventStorming* na fase de *design* e o *Behavior-Driven Development (BDD)* nas etapas de requisitos e testes, visando abordar vulnerabilidades de forma proativa.

O ciclo inicia-se com um *Workshop* de *EventStorming*. Nesta etapa colaborativa, a equipe realiza a Descoberta dos fluxos de eventos e interações entre os serviços, permitindo uma análise aprofundada da arquitetura e a identificação de pontos de exposição, falhas potenciais e a superfície de ataque do sistema. A Descoberta, por sua vez, resulta na elaboração de Recomendações baseadas em padrões de segurança, que consistem em propostas de melhoria arquitetônica para mitigar as vulnerabilidades identificadas.

Tais recomendações são formalizadas na etapa seguinte, a Adoção de *BDD*. Nela, as ideias de segurança são traduzidas em Histórias de usuários com critérios de aceitação específicos para requisitos não funcionais. Esses artefatos servem como guias para a equipe de desenvolvimento, direcionando a implementação de funcionalidades com as devidas considerações de segurança.

Por fim, o ciclo é concluído com a Automação de Testes. Os critérios de aceitação definidos no *BDD* se tornam a base para a criação de testes automatizados que validam a correta implementação das medidas de segurança planejadas. Essa automação visa garantir continuamente a conformidade do sistema com os requisitos de segurança e assegurar que os problemas identificados tenham sido sanados. Dessa forma, o modelo garante que a segurança não seja um *afterthought*, mas um processo contínuo e sistemático, integrado desde a análise até a verificação final da solução.

Figura 4 – Ciclo de Análise de Segurança para Arquitetura de Microsserviços



4.3 Relatos de experiência

Este relato descreve uma vivência relevante que visa contribuir tanto para o campo de atuação do pesquisador quanto para outros profissionais da área. Os resultados apresentados possuem potencial de generalização, servindo como exemplos aplicáveis a situações semelhantes e subsidiando estudos futuros. A relevância de um relato de experiência está na identificação de problemas significativos, bem como na possibilidade de aplicar os procedimentos e resultados obtidos em contextos correlatos. Dessa forma, constitui uma contribuição substancial para o aprimoramento das práticas metodológicas no campo de estudo.

O objeto deste relato é um sistema de *e-commerce open source* estruturado em arquitetura de microsserviços. O principal objetivo é analisar a superfície de ataque desse software utilizando a abordagem *EventStorming*, com o intuito de identificar potenciais vulnerabilidades. Para tanto, foi realizado um mapeamento detalhado dos eventos do sistema, permitindo compreender as interações entre seus componentes. Conforme definido por (BRANDOLINI, 2013), os eventos são identificados e organizados preferencialmente em ordem cronológica, o que proporciona uma visão sequencial dos processos e facilita a detecção de pontos frágeis no fluxo de execução do sistema.

Ao defender a importância da sistematização de processos, métodos, técnicas e ferramentas para assegurar o sucesso no desenvolvimento de software com produtividade e qualidade, a Engenharia de Software consolida seu lugar no campo da Computação. Paralelamente, a área de Sistemas de Informação dedica-se a analisar a composição dos elementos envolvendo tecnologias, processos, pessoas e organizações, investigando a dinâmica dessas interações, seus efeitos, *feedbacks* e evolução enquanto sistemas complexos. A metodologia empregada nesta pesquisa aplicação do *EventStorming* para análise da superfície de ataque

em sistemas com arquitetura de microsserviços foi conduzida por meio de um estudo de caso no software livre *Spring Cloud Event Sourcing Example*. O objetivo foi avaliar a aplicabilidade do artefato Big Picture, gerado durante o Event Storming, para a análise da superfície de ataque.

Este relato busca analisar uma situação desafiadora, com o propósito de compreender e produzir conhecimento que possa fundamentar teoricamente avaliações em contextos semelhantes. A investigação caracteriza-se como exploratória e descritiva, pois forneceu informações úteis para identificar vulnerabilidades potenciais. Entretanto, o domínio estudado requer dados adicionais para garantir a validade e confiabilidade das conclusões, mediante triangulação e confirmação dos fatos.

4.4 *EventStorming* na Análise de Superfície de Ataque

Por meio da utilização do *EventStorming*, realizamos o mapeamento dos eventos do sistema. Esses eventos, conforme definido por Brandolini (BRANDOLINI, 2013), são identificados e organizados preferencialmente em ordem cronológica. Em seguida, associamos aos eventos os comandos que os disparam, de modo que os próprios eventos e as políticas incluindo condições, decisões e gatilhos funcionem como ponte entre eventos distintos. Logo após, atribuímos aos comandos os usuários que os acionam. Este é o primeiro ponto de atenção do estudo, pois os usuários dos comandos indicam que esses comandos estão potencialmente expostos.

Nem todos os comandos são iniciados diretamente por um usuário; são esses comandos que utilizaremos para identificar a interface de acesso aos microsserviços estabelecidos. Ao concluir essa etapa, quando todos os *post-its* formam a *Big Picture* do sistema, são atribuídos os contextos que, em última análise, correspondem aos próprios microsserviços. Os comandos acessados por usuários, sejam eles autenticados ou não, aumentam naturalmente a superfície de ataque.

Considerando que o conceito de microsserviços prevê limites bem definidos e encapsulados (RESELMAN, 2022a), o caminho natural seria que cada microsserviço gerencie sua própria segurança de acesso. Contudo, mesmo em sistemas de pequeno porte, essa arquitetura enfrenta desafios ao tentar implementar uma camada de segurança distinta em cada microsserviço.

4.4.1 Visão Geral

Com todos os *post-its* organizados, torna-se possível visualizar a interação entre os diferentes contextos do sistema, conforme ilustrado na Figura 5.

O *EventStorming*, um *workshop* desenvolvido por Alberto Brandolini (BRANDOLINI, 2013) e uma adaptação do *Domain-Driven Design* (RADEMACHER; SORGALLA;

SACHWEH, 2018), é uma ferramenta colaborativa que resulta em uma representação visual focada no comportamento do software. Sua capacidade de integrar a visão técnica detalhada com uma perspectiva mais abstrata a *Big Picture* o torna eficaz para o entendimento compartilhado de domínios complexos.

Essa metodologia é valiosa na análise de projetos existentes, facilitando a identificação e visualização dos eventos de negócio de um sistema. Sua aplicação, inclusive em processos de engenharia reversa, como a realizada no sistema open source Spring Cloud Event Sourcing Example, visa obter uma visão sistêmica de alto nível do funcionamento do domínio, abrangendo jornadas de usuários, fluxos de trabalho e processos de negócio. A *Big Picture*, nesse contexto, transcende aspectos tecnológicos, considerando também a cultura de produto e o controle de design para soluções mais completas e eficientes.

Tradicionalmente utilizado para modelagem de domínios empresariais, o *EventStorming* pode ser adaptado para análises de segurança. Durante sua aplicação, a participação de um especialista em segurança enriquece a identificação de ações e eventos do sistema, permitindo destacar pontos potencialmente vulneráveis. Contudo, uma limitação dessa abordagem é sua natureza menos formal, que pode dificultar a documentação precisa de aspectos de segurança, especialmente em sistemas mais complexos.

4.4.2 Adaptações

A adaptação do *EventStorming* para a análise da superfície de ataque exigiu uma reconfiguração de seu propósito original, tradicionalmente voltado à modelagem de domínios de negócio. Essa adaptação visou possibilitar a identificação de pontos de exposição e vulnerabilidades no sistema. Nesse contexto, a metodologia foi aplicada por meio de um processo de engenharia reversa em uma loja virtual, permitindo o mapeamento dos *endpoints* expostos e das interações entre usuários e sistema, o que proporcionou uma visão estruturada da superfície de ataque.

A priorização dos eventos relacionados à segurança tornou-se um aspecto essencial dessa adaptação, possibilitando a identificação das áreas críticas onde a implementação de mecanismos de proteção, como autenticação e criptografia, é fundamental para mitigar riscos. A adoção de uma abordagem iterativa mostrou-se crucial para a revisão contínua do modelo de segurança, garantindo sua adequação às modificações na arquitetura do sistema.

Todavia, algumas limitações dessa adaptação não foram abordadas no artigo, como a subjetividade inerente à interpretação dos eventos, a necessidade de conhecimento especializado em segurança para a realização de uma análise eficaz, e os desafios decorrentes da aplicação da técnica em sistemas altamente dinâmicos e com múltiplas integrações externas.

Identificação dos pontos de Exposição

A aplicação do *EventStorming* transcende o escopo tradicional da Engenharia de Requisitos, mostrando-se eficaz na análise e aprimoramento de sistemas já implementados. No contexto deste relato, a técnica foi adaptada para realizar engenharia reversa em uma loja virtual consolidada, o que permitiu a identificação precisa de *endpoints* e interfaces expostas. Essa abordagem revelou-se estratégica para evidenciar pontos de exposição ao longo do fluxo de dados e das interações do sistema, proporcionando uma visão abrangente das áreas onde potenciais vulnerabilidades podem surgir, conforme ilustra a Figura 6.

Esse processo visualiza transações e eventos entre os componentes da loja online, facilitando a detecção de pontos críticos e a análise detalhada das interações entre usuários e sistema. Tal capacidade é particularmente valiosa em sistemas complexos, onde o volume elevado de interações aumenta a suscetibilidade a falhas de segurança. Ao mapear essas interações e áreas de exposição, a metodologia fornece subsídios para decisões fundamentadas sobre proteções adequadas em cada ponto de entrada, como autenticação, criptografia e uso de *firewalls*. Dessa forma, o *EventStorming* consolida-se como uma técnica versátil e adaptável, que não apenas facilita o entendimento do fluxo de trabalho de um sistema implementado, mas também possibilita uma análise de segurança colaborativa e direcionada, essencial para a manutenção da integridade e da confiabilidade em ambientes de *e-commerce*.

Foco em eventos de segurança

A crescente expansão do *e-commerce* solidifica-o como uma alternativa atrativa e cada vez mais comum para consumidores e empreendedores. No entanto, essa tendência traz desafios significativos, sendo a segurança do ambiente online um aspecto crítico. A percepção de insegurança por parte do usuário pode levar ao abandono do carrinho ou à hesitação na finalização da transação, impactando diretamente o desempenho e a rentabilidade do negócio.

Nesse cenário, metodologias para identificação e mitigação de vulnerabilidades tornam-se estratégicas. O *EventStorming*, embora amplamente aplicado na Engenharia de Requisitos, pode ser expandido para a análise de segurança em sistemas existentes. Por meio da engenharia reversa, essa técnica visual de mapeamento de eventos é aplicada em uma loja online consolidada para identificar, de forma estruturada, *endpoints* e pontos de integração expostos ao usuário, possibilitando o mapeamento da superfície de ataque do sistema.

A utilização do *EventStorming* no contexto do *e-commerce* amplia o entendimento sobre o tráfego de informações na aplicação, evidenciando potenciais pontos de vulnerabilidade. A técnica promove uma compreensão colaborativa das interações do sistema entre desenvolvedores, analistas de segurança e gestores, o que facilita a identificação de áreas onde mecanismos de segurança, como autenticação e criptografia de dados, podem ser aplicados eficazmente.

A aplicação prática do *EventStorming* em análises de segurança converte um processo originalmente voltado ao desenvolvimento de novos sistemas em um recurso valioso para a

avaliação de segurança de sistemas já existentes, como lojas online. Essa abordagem não apenas protege os dados dos usuários, mas também solidifica a confiança do consumidor — aspecto crucial para o êxito em plataformas de comércio eletrônico. Em suma, a integração de engenharia reversa e metodologias visuais como o *EventStorming* robustece a segurança do *e-commerce*, contribuindo para ambientes digitais seguros e confiáveis que incentivam a transação online.

Revisão e Iteração

A superfície de ataque de um sistema digital não é estática, sofrendo alterações significativas com atualizações, novas funcionalidades e modificações arquiteturais. Nesse contexto, a revisão e atualização periódicas do modelo de segurança são essenciais para refletir a configuração atual do sistema, possibilitando a identificação e mitigação proativa de novas vulnerabilidades.

Um ciclo contínuo de revisão e iteração permite que as equipes de desenvolvimento e segurança reavaliem constantemente interações e *endpoints* expostos. Isso assegura uma proteção proativa contra ameaças, pois mudanças no sistema como a inclusão de novos serviços ou alterações nos fluxos de dados podem criar novos pontos vulneráveis.

Essa prática sistemática facilita a detecção precoce de vulnerabilidades potenciais e o fortalecimento das defesas existentes, contribuindo significativamente para a resiliência do sistema. Adicionalmente, oferece uma visão atualizada do estado de segurança, crucial para o cumprimento de normas e regulamentações. Dessa forma, a iteração regular do modelo de segurança é uma prática indispensável para assegurar a integridade e a confiabilidade em ambientes digitais dinâmicos e sujeitos a constantes mudanças.

4.4.3 Design do relato de experiência

A elaboração deste relato de experiência e a sistematização de processos, métodos e técnicas visam contribuir para o desenvolvimento de software com produtividade e qualidade em ambientes complexos. Nesse contexto, a Engenharia de Software e a área de Sistemas de Informação atuam na análise da dinâmica entre tecnologias, processos, pessoas e organizações, e sua evolução como sistemas complexos.

O design desta pesquisa fundamenta-se na aplicação do *EventStorming* para a análise da superfície de ataque em sistemas com arquitetura de microsserviços. Conduzida como um estudo de caso sobre o software livre *Spring Cloud Event Sourcing Example*, esta investigação tem como objetivo avaliar se o artefato *Big Picture*, gerado durante o *EventStorming*, pode servir como suporte para tal análise.

Caracterizando-se como exploratória e descritiva, a investigação busca compreender uma situação desafiadora e gerar conhecimento que sirva de base teórica para a avaliação de contextos semelhantes, fornecendo informações relevantes para a identificação de

possíveis vulnerabilidades. Contudo, reconhece-se que o domínio em estudo demanda dados adicionais para assegurar a validade e confiabilidade das informações, o que requer futura triangulação e confirmação dos fatos.

4.4.4 Eficácia

A aplicação do *EventStorming* demonstra-se eficaz para a análise da superfície de ataque em arquiteturas de microsserviços por múltiplos fatores. Primeiramente, sua representação visual de eventos e fluxos de comunicação facilita um mapeamento preciso das fronteiras dos microsserviços e das interações entre eles, aspecto fundamental para a definição clara de domínios e responsabilidades. Essa clareza arquitetural, aliada à capacidade de segmentar eventos por contexto de negócio, naturalmente alinha o desenho do sistema com os princípios de coesão e responsabilidade única.

Adicionalmente, a visualização dos eventos e interações permite à equipe identificar proativamente potenciais vulnerabilidades e pontos críticos da superfície de ataque, como chamadas externas e mecanismos de autenticação/autorização. Tal discernimento facilita a discussão e o planejamento de estratégias de mitigação de riscos, incluindo a implementação de autenticação *JWT*, o uso obrigatório de HTTPS e a configuração de *gateways* seguros para a troca de dados. Esta camada extra de compreensão compartilhada dos riscos e soluções contribui significativamente para a segurança.

Por fim, a natureza visual e colaborativa do *EventStorming* é crucial em arquiteturas de microsserviços, que demandam interação ativa entre equipes multidisciplinares. Essa abordagem promove uma compreensão coletiva da estrutura e dos pontos críticos de segurança, reduzindo mal-entendidos e aprimorando a percepção de riscos. Sendo ágil e baseada em *post-its*, a metodologia permite ajustes rápidos e contínuos no modelo arquitetural e de segurança, refletindo a dinâmica do ambiente de negócios e garantindo o alinhamento constante com a evolução do sistema.

4.5 Estudo de Caso

A Engenharia de Software Baseada em Evidência é uma metodologia cujo propósito é promover o desenvolvimento de sistemas com maior confiabilidade (KITCHENHAM; DYBA; JORGENSEN, 2004), por meio, por exemplo, da condução de estudos de caso. O uso de estudos de caso tem se expandido consideravelmente na área de Engenharia de Software (MELEGATI; WANG, 2020), pois permite a coleta de dados empíricos voltados à compreensão de uma população específica, com o objetivo de subsidiar o desenvolvimento de sistemas mais confiáveis. Assim, a Engenharia de Software fundamenta-se na observação das ações relacionadas aos softwares e suas respectivas respostas

(WOHLIN; HÖST; HENNINGSSON, 2003), permitindo a validação das expectativas em relação ao comportamento do sistema.

Esta pesquisa classifica-se como exploratória e descritiva (RUNESON; HÖST, 2009; YIN, 2012), dada a necessidade inerente de identificar requisitos não funcionais por meio de *frameworks* ágeis, com foco na integração da validação da qualidade. Foram seguidas diretrizes propostas por Melegati e Wang (MELEGATI; WANG, 2020) e por Petersen (PETERSEN, 2020), que incluem a definição clara dos objetivos e das questões de pesquisa, a seleção apropriada de casos relevantes, além da coleta e análise de dados provenientes de múltiplas fontes, como entrevistas, observações e documentos. Também se buscou garantir a validade e a confiabilidade dos dados por meio da triangulação e da verificação com os participantes do estudo.

Este trabalho compreende dois estudos de caso distintos. O primeiro investiga a aplicação da abordagem *Behavior-Driven Development (BDD)* para o tratamento de requisitos não funcionais e a automação de testes, com ênfase na característica de eficiência de desempenho, conforme estabelecido pela *norma ISO/IEC/IEEE 25010*. Esse estudo foi conduzido em colaboração com uma empresa de desenvolvimento de software, analisando uma equipe experiente na adoção do *BDD* para fins de automação de testes e definição de requisitos não funcionais.

O segundo estudo de caso adota a metodologia da Engenharia de Software Baseada em Evidência (*Evidence-Based Software Engineering*), com o objetivo de aumentar a confiabilidade dos sistemas desenvolvidos (KITCHENHAM; DYBA; JORGENSEN, 2004). A pesquisa analisou um experimento que envolveu a aplicação do *BDD* para requisitos não funcionais e automação de testes, considerando os requisitos de segurança definidos na *norma ISO/IEC 25010:2023*, com o apoio de soluções baseadas em inteligência artificial generativa.

4.5.1 Uso de Behavior-Driven Development (BDD) para Requisitos Não Funcionais

A metodologia adotada nesta pesquisa investiga a aplicação do *Behavior-Driven Development (BDD)* (SANTOS et al., 2024a) para o tratamento de requisitos não funcionais e automação de testes, com base na característica de eficiência de desempenho definida pela norma *ISO/IEC/IEEE 25010*. O estudo foi conduzido por meio de um caso em colaboração com uma empresa desenvolvedora de software.

Desenho da Pesquisa

Esta pesquisa tem como objetivo explorar e descrever fenômenos relacionados à obtenção de requisitos não funcionais por meio de *frameworks* ágeis, visando à integração da validação da qualidade (RUNESON; HÖST, 2009; YIN, 2012). As orientações propostas

por Melegati e Wang (MELEGATI; WANG, 2020) e Petersen (PETERSEN, 2020) foram seguidas, envolvendo a definição clara dos objetivos e das questões de pesquisa, a seleção criteriosa de casos relevantes, e a coleta e análise de dados provenientes de múltiplas fontes, como entrevistas, observações e documentação. Buscou-se assegurar a validade e a confiabilidade dos dados por meio da triangulação metodológica e da verificação pelos participantes *(member checking)*.

Protocolo de Pesquisa e Questões

O protocolo empregado para direcionar este estudo de caso foi formalizado usando parte do modelo *Goal-Question-Metric* (WOHLIN et al., 2012), definido como: "**Analisar** a aplicação do BDD, **com finalidade de** automatizar, **em relação a** requisitos não-funcionais baseados nas características da Norma ISO/IEC/IEEE 25010, **do ponto de vista** dos envolvidos no produto de software, **no contexto da** característica de desempenho". A fim de alcançar o objetivo proposto, foram elaboradas as questões de pesquisa expressas na Tabela 9.

Tabela 9 – Questões de Pesquisa

Questão	Descrição	Justificativa
QP1	Quais as características relacionadas à qualidade abordadas na Norma ISO/IEC/IEEE 25010?	O objetivo desta questão foi identificar os aspectos que seriam considerados na elicitação dos requisitos não-funcionais.
QP2	Como o BDD aborda as dificuldades em relação à garantia dos requisitos não-funcionais?	O objetivo desta questão foi entender como o BDD consegue mitigar as situações advindas da elicitação de requisitos não-funcionais.
QP3	Como o BDD pode ser usado para garantir a qualidade relacionada a característica de eficiência de desempenho da Norma ISO/IEC/IEEE 25010?	O objetivo desta questão foi explanar como este *framework* poderia contribuir na elicitação de requisitos não-funcionais.
QP4	Quais são os benefícios de usar o BDD em termos de identificação e automação dos testes não-funcionais?	O objetivo desta questão foi trazer os pontos positivos voltados à adoção do BDD no ciclo de testes.

Descrição do Caso de Estudo

A análise foi realizada com uma equipe experiente que já utilizava o *Behavior-Driven Development (BDD)* no tratamento de requisitos não funcionais e na automação de testes,

com foco na característica de eficiência de desempenho da norma *ISO/IEC/IEEE 25010*. Essa característica refere-se à capacidade do produto de software em fornecer desempenho apropriado em relação ao uso de recursos e ao tempo de resposta sob condições específicas. Ela é subdividida em três subcaracterísticas principais:

- Comportamento em relação ao tempo;
- Utilização de recursos;
- Capacidade.

Execução A execução do estudo de caso foi realizada com base nas funcionalidades e nos cenários definidos conforme a norma *ISO/IEC/IEEE 25010*, tendo como objetivo avaliar a eficiência de desempenho em relação às três subcaracterísticas dessa dimensão. A avaliação seguiu a linguagem *Gherkin*, por meio da descrição de comportamentos observáveis, alinhados aos requisitos não funcionais especificados.

História de usuário: Eficiência do sistema (comportamento em relação ao tempo)

Descrição: Como administrador do sistema, a equipe busca garantir que o sistema suporte 100 usuários simultâneos por um período contínuo de sete dias, assegurando que a eficiência se mantenha consistente durante períodos de alta demanda.

Cenário: Comportamento em relação ao tempo

Dado que o sistema está funcionando corretamente e tem capacidade para 100 usuários simultâneos, conforme Figura 7.

Quando 100 usuários simultâneos acessarem o sistema por 7 dias consecutivos e cada usuário criar uma oportunidade a cada 5 minutos, conforme Figura 8.

Então o sistema deve manter um tempo de resposta médio de menos de 5 segundos, não deve apresentar erros críticos durante o teste e a utilização de recursos do servidor deve permanecer abaixo de 80%, conforme Figura 9.

História de usuário: Utilização de recursos

Descrição: Como administrador de sistema, a equipe busca garantir que o sistema possa lidar com 1000 usuários simultâneos por 30 minutos, assegurando que a utilização de recursos do servidor permaneça abaixo de 85% durante o teste de carga.

Cenário: Teste de utilização de recursos

Dado que o sistema está funcionando corretamente e tem capacidade para 1000 usuários simultâneos.

Quando 1000 usuários simultâneos acessarem o sistema por 30 minutos.

Então a utilização de recursos do servidor não deve exceder 85% e o sistema não deve apresentar erros críticos durante o teste.

História de usuário: Capacidade de usuários

Descrição: Como administrador de sistema, a equipe busca garantir que o sistema possa lidar com 2.700 usuários simultâneos por 30 minutos, assegurando que a capacidade do sistema seja suficiente para atender à demanda durante picos de uso.

Cenário: Teste de capacidade do sistema

Dado que o sistema está funcionando corretamente e tem capacidade para 2.700 usuários simultâneos.

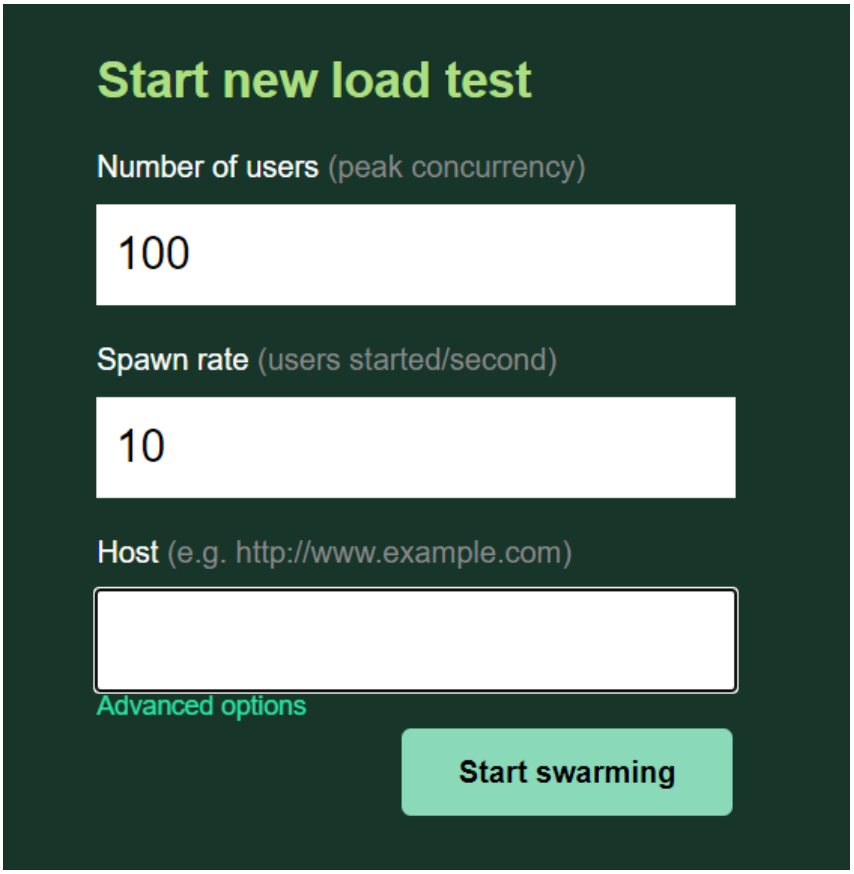
Quando 2.700 usuários simultâneos acessarem o sistema por 30 minutos.

Então a utilização de recursos do servidor não deve exceder 90% e o sistema não deve apresentar erros críticos durante o teste.

Configuração e execução

A execução desses cenários de teste permitiu avaliar o desempenho do sistema em relação às subcaracterísticas mencionadas e garantir que sejam atendidos os requisitos não-funcionais definidos. Assim, foram elaborados *scripts* para testes no Python com Locust. Ao executar, as configurações para cada cenário de usuários e execução são feitas na tela, conforme Figura 10.

Figura 10 – Configurando o Locust



4.5.2 Um experimento com foco em segurança por meio de Modelos de Linguagem de Grande Escala usando Desenvolvimento Orientado a Comportamento

Este estudo investigou a eficácia da geração textual por inteligência artificial por meio de prompts direcionados à manutenção da qualidade no desenvolvimento de software, com ênfase na economia de tempo na elicitação de requisitos e na automação de testes. O experimento foi conduzido com base em funcionalidades e cenários alinhados à norma *ISO/IEC 25010:2023*, utilizando técnicas de *BDD* e inteligência artificial generativa. A análise concentrou-se no requisito não funcional de segurança e em suas seis subcaracterísticas: confidencialidade, integridade, não repúdio, responsabilização, autenticidade e resistência. O processo de elaboração dos prompts para a elicitação automatizada desses requisitos está descrito na Figura 11, e os resultados correspondentes encontram-se disponíveis para consulta[1].

Para cada subcaracterística analisada neste estudo, foi definida uma instrução específica, com o intuito de atender às particularidades de cada requisito não funcional. As informações complementares foram organizadas em conformidade com as demandas específicas de

[1] Omitido para revisão cega.

cada subcaracterística, otimizando assim sua utilização em modelos de linguagem de grande escala *(LLMs)*. Quando pertinente, os resultados gerados por esses modelos foram refinados para assegurar a qualidade e a adequação das respostas obtidas a partir das instruções elaboradas.

4.5.3 Execução do experimento

O modelo adotado para a condução deste experimento está fundamentado na proposta de Rajbhoj et al. (RAJBHOJ et al., 2024), conforme ilustrado na Figura 12.

O experimento descrito neste artigo teve como foco a avaliação da eficácia da geração textual por inteligência artificial, por meio de *prompts* elaborados com o objetivo de manter a qualidade no desenvolvimento de software. A proposta visa, ainda, otimizar o tempo dedicado à elicitação de requisitos e à automação de testes. A execução do experimento fundamentou-se em funcionalidades e cenários definidos conforme a norma *ISO/IEC 25010:2023*, utilizando a abordagem de Desenvolvimento Orientado por Comportamento (*BDD*) e modelos de linguagem generativa.

4.5.4 Métrica Avaliada

A métrica central avaliada neste estudo é a similaridade, que se refere à comparação entre os resultados gerados pelos modelos de linguagem *ChatGPT (GPT-4o)* e *Gemini (gemini-1.5-flash-8b-001)*. O objetivo consiste em identificar a existência de correspondência ou divergência significativa entre as respostas produzidas por esses *LLMs*. Por meio da análise estatística dos dados obtidos, busca-se fornecer resultados consistentes e fundamentados, capazes de sustentar conclusões embasadas acerca do desempenho comparativo entre os modelos avaliados.

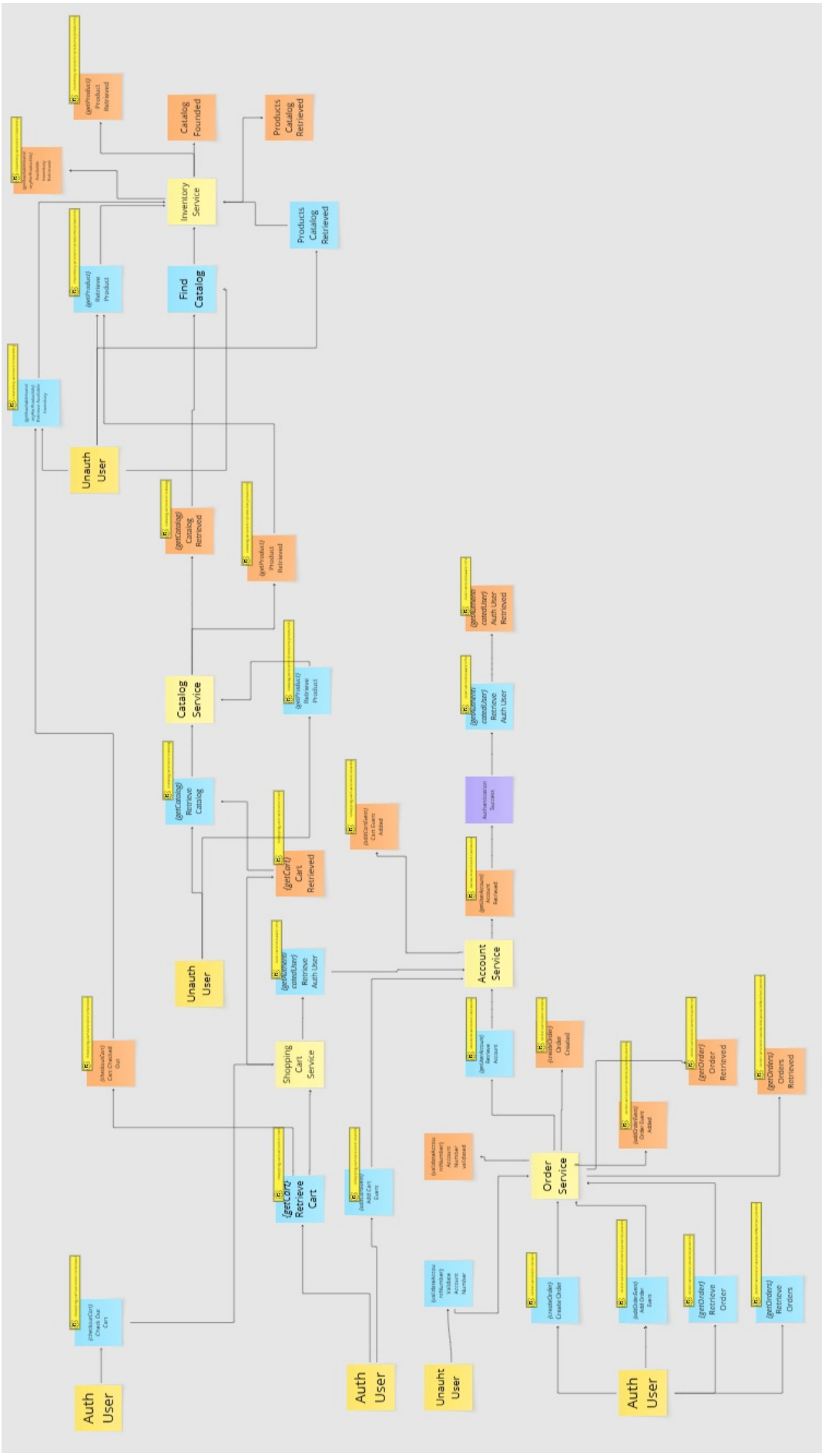


Figura 5 – Exemplo de uma figura geral obtida por meio de *EventStorming*

Figura 6 – Metodologia Utilizada

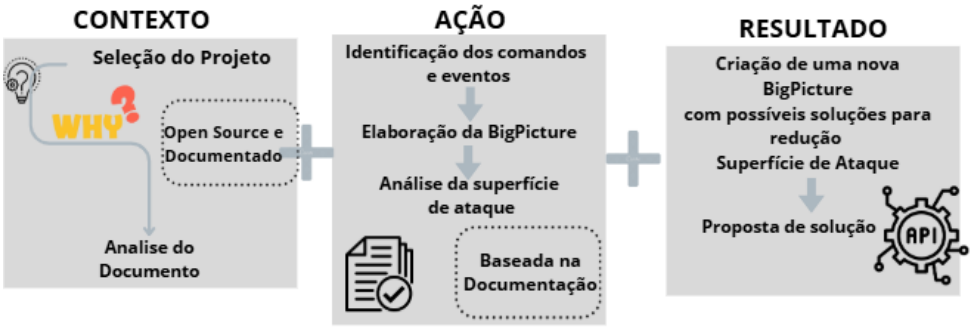


Figura 7 – Número de usuários

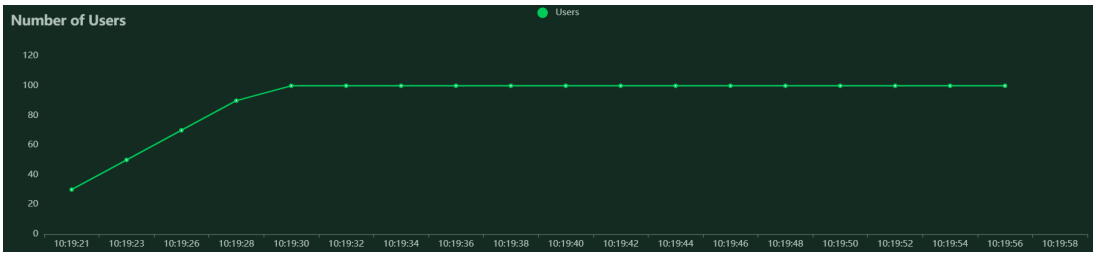


Figura 8 – Utilização da CPU e da memória

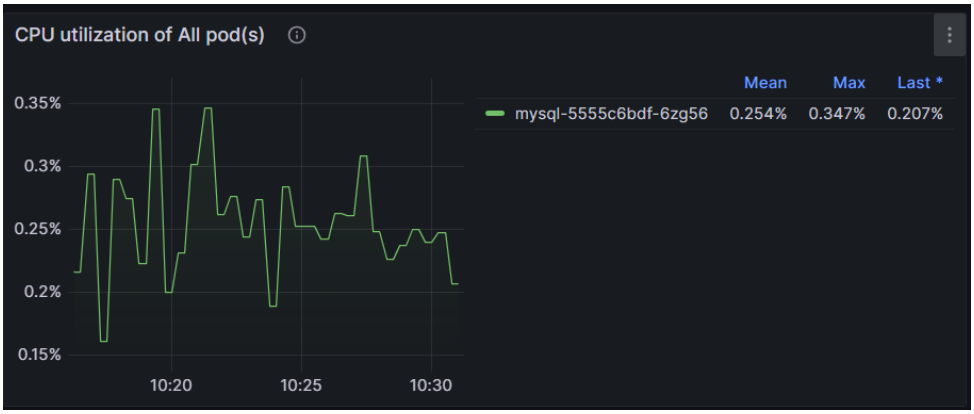


Figura 9 – Tempo de resposta

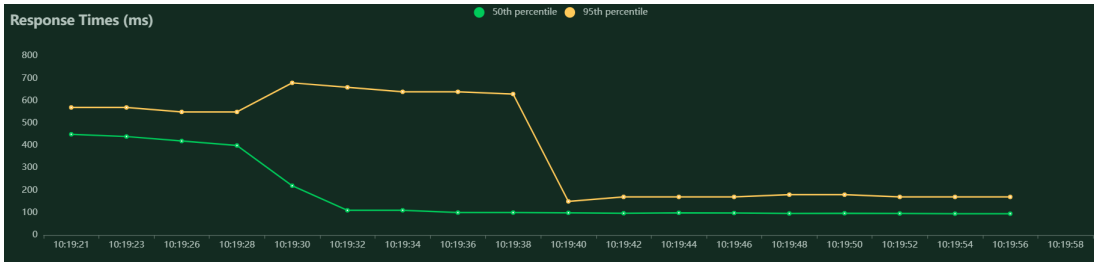


Figura 11 – Processo de automatização de Prompt

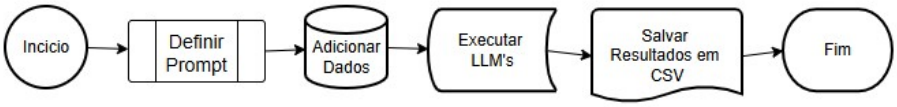
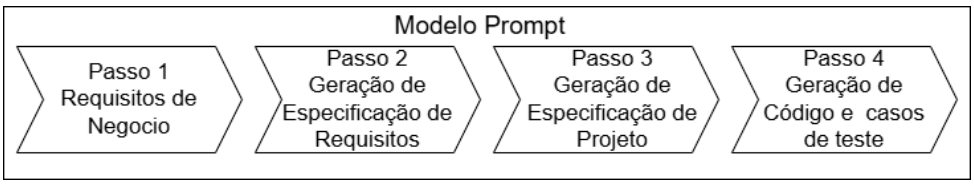


Figura 12 – Processo de ciclo de vida de desenvolvimento de software usando IA generativa [adaptado(RAJBHOJ et al., 2024)]



5
Resultados

Este capítulo detalha os resultados obtidos por meio da metodologia de pesquisa descrita no Capítulo 4. A apresentação dos achados está estruturada em três frentes de investigação distintas: o Mapeamento Sistemático de Literatura (MSL), que fundamenta a base teórica; os relatos de experiência, que abordam a aplicação prática do *EventStorming* para análise de superfícies de ataque e do *Behavior-Driven Development (BDD)* para requisitos não-funcionais; e, por fim, o experimento conduzido com Modelos de Grande Linguagem *(LLMs)* no contexto da segurança de software.

5.1 Mapeamento Sistemático de Literatura (MSL)

A etapa de Mapeamento Sistemático de Literatura foi conduzida conforme detalhado no Capítulo 4. Inicialmente, a função de busca automática da ferramenta *Parsifal* foi utilizada para identificar duplicatas, resultando na remoção de 129 artigos, correspondentes a 97% do total inicial. Posteriormente, realizou-se a triagem dos títulos e resumos, considerando sua relevância em relação às questões de pesquisa. A técnica de *snowballing* também foi aplicada para ampliar a abrangência do mapeamento. Cada estudo foi classificado de forma independente como aceito ou rejeitado, com base nos critérios apresentados na Tabela 8. Após essa etapa de triagem e seleção, 45 artigos foram eliminados por não ser possível o acesso ao texto completo para leitura na íntegra. Os estudos selecionados foram então submetidos à leitura completa para extração de dados. Essa abordagem sistemática assegurou uma seleção criteriosa dos estudos, aumentando a confiabilidade dos resultados. O processo de extração está ilustrado na Figura 3.

Os estudos que efetivamente contribuem para a resposta às questões de pesquisa estão listados na Tabela 10.

Tabela 10 – Artigos Primários Selecionados no Mapeamento Sistemático da Literatura

Artigo (Referência)	**Título**	**Contribuição para as RQPs**
(WANG; KADIYALA; RUBIN, 2021)	Promises and challenges of microservices: an exploratory study	RQP1, RQP2, RQP3, RQP5
(YIN; DU, 2020)	On representing resilience requirements of microservice architecture systems	RQP2
(BANGUI; ROSSI; BUHNOVA, 2022)	A conceptual antifragile microservice framework for reshaping critical infrastructure	RQP1, RQP3, RQP4
(XU et al., 2023a)	Zero-trust security authentication based on spa and endogenous security architecture	RQP1, RQP3, RQP5
(ANTUNES, 2024)	Evaluating intrusion detection for microservice applications: Benchmark, dataset, and case studies	RQP3
(PAI; HEBBAR; KUMAR, 2023)	Securing Microservice-Driven Applications Based on API Access Graphs Using Supervised Machine Learning Techniques	RQP1
(MATIAS et al., 2024)	Evaluating Effectiveness and Security in Microservices Architecture	RQP1, RQP2, RQP3, RQP4
(ALKADI; MOUSTAFA; TURNBULL, 2020)	A Review of Intrusion Detection and Blockchain Applications in the Cloud: Approaches, Challenges and Solutions	RQP2, RQP5

5.1.1 Análise e Síntese dos Achados do MSL

O Mapeamento Sistemático de Literatura apresentado teve como objetivo analisar a superfície de ataque em sistemas distribuídos, com ênfase em arquiteturas baseadas em microsserviços, visando identificar os principais desafios e vulnerabilidades desse contexto. Por meio de uma abordagem teórico-aplicada, foram evidenciadas lacunas nas estratégias de segurança atualmente empregadas, destacando-se a necessidade de protocolos mais robustos para a mitigação de riscos. Utilizando metodologias sistemáticas, com o apoio da ferramenta *Parsifal*, o mapeamento investigou métodos de pesquisa voltados à redução da superfície de ataque. Os achados reforçam a importância de abordagens estruturadas, contínuas e multidisciplinares para o fortalecimento da resiliência e da integridade de sistemas complexos. Ademais, o mapeamento sistemático delineia direções promissoras para futuras investigações no campo da segurança em ambientes de microsserviços.

A análise da superfície de ataque em sistemas baseados em microsserviços requer uma abordagem colaborativa e iterativa que integre múltiplas técnicas e práticas consolidadas na engenharia de software segura. Nesse contexto, propõe-se um ciclo estruturado em três

etapas principais: A primeira etapa consiste na realização de *workshops* de *EventStorming*, que permitem a descoberta dos fluxos de eventos e interações entre os componentes do sistema, evidenciando pontos de exposição potencial à ataques. Essa técnica viabiliza a identificação de comandos, eventos, atores e agregadores, elementos essenciais para mapear a superfície de ataque e gerar recomendações baseadas em padrões de segurança reconhecidos. A segunda etapa do ciclo envolve a adoção da abordagem *Behavior-Driven Development (BDD)*, com foco específico na elicitação de requisitos não funcionais. A partir dos *insights* obtidos na fase anterior, são redigidas histórias de usuários com critérios de aceite voltados para aspectos como confidencialidade, disponibilidade, integridade e resiliência do sistema. Essa formalização facilita a rastreabilidade dos requisitos de segurança ao longo do ciclo de vida do software, promovendo alinhamento entre times de desenvolvimento, qualidade e segurança da informação. Por fim, a terceira etapa corresponde à automação dos testes derivados das histórias de *BDD*. Essa automação permite validar continuamente os critérios de segurança estabelecidos, garantindo conformidade com os objetivos definidos nas fases anteriores. Além disso, contribui para a manutenção de um pipeline de entrega contínua que incorpora segurança desde os estágios iniciais do desenvolvimento. Com isso, o ciclo proposto não apenas cobre as fases de análise e modelagem, mas também assegura a verificação contínua e sistemática da segurança em ambientes distribuídos com arquitetura de microsserviços. Porém, citando as suas descobertas no mapeamento sistemático.

A análise da superfície de ataque em sistemas baseados em microsserviços exige uma abordagem colaborativa e iterativa que integre múltiplas técnicas e práticas consolidadas da engenharia de software segura. Nesse contexto, propomos um ciclo estruturado em três etapas principais:

- A primeira etapa concentra-se na realização de *Workshops* de *EventStorming*. Essa técnica permite a descoberta e o mapeamento dos fluxos de eventos e interações entre os componentes do sistema, revelando pontos potenciais de exposição a ataques. A identificação de comandos, eventos, atores e agregados durante sua aplicação é crucial para a análise da superfície de ataque, pois esses elementos são fundamentais para gerar recomendações alinhadas a padrões de segurança reconhecidos.
- A segunda etapa consiste na aplicação do *Behavior-Driven Development (BDD)*, com foco primordial na elicitação de requisitos não funcionais. Fundamentado nos *insights* provenientes do *EventStorming*, este estágio envolve a redação de histórias de usuários cujos critérios de aceite abordam aspectos críticos como confidencialidade, disponibilidade, integridade e resiliência do sistema. Essa formalização impulsiona a rastreabilidade dos requisitos de segurança ao longo do ciclo de vida do software, assegurando o alinhamento efetivo entre as equipes de desenvolvimento, qualidade e segurança da informação.

- A etapa final consiste na automação dos testes derivados das histórias de *BDD*. Essa automação permite a validação contínua dos critérios de segurança previamente estabelecidos, garantindo a conformidade com os objetivos definidos nas fases anteriores. Além disso, ela contribui significativamente para a manutenção de um *pipeline* de entrega contínua que integra a segurança desde os estágios iniciais do desenvolvimento.

A proposta metodológica delineada para a análise da superfície de ataque e elicitação de requisitos em microsserviços não se restringe às fases de análise e modelagem. Ela se estende à verificação contínua e sistemática da segurança em ambientes distribuídos, fundamentada na arquitetura de microsserviços.

As diretrizes do ciclo proposto são corroboradas pelas descobertas do nosso mapeamento sistemático de literatura. Os resultados desse mapeamento destacam estratégias-chave para a redução de vulnerabilidades, ao mesmo tempo em que revelam lacunas críticas nas práticas de segurança atuais. Adicionalmente, fornecem *insights* acionáveis tanto para pesquisadores quanto para profissionais da área. Em conjunto, essas descobertas contribuem significativamente para o desenvolvimento de sistemas web mais resilientes, oferecendo uma orientação estruturada para a implementação de um design focado em segurança em arquiteturas distribuídas.

5.2 Relato de Experiência

Esta seção apresenta os resultados obtidos a partir dos relatos de experiência conduzidos, que abordaram a aplicação do *EventStorming* para análise de superfícies de ataque e o uso do *Behavior-Driven Development (BDD)* para a especificação e verificação de requisitos não-funcionais de desempenho.

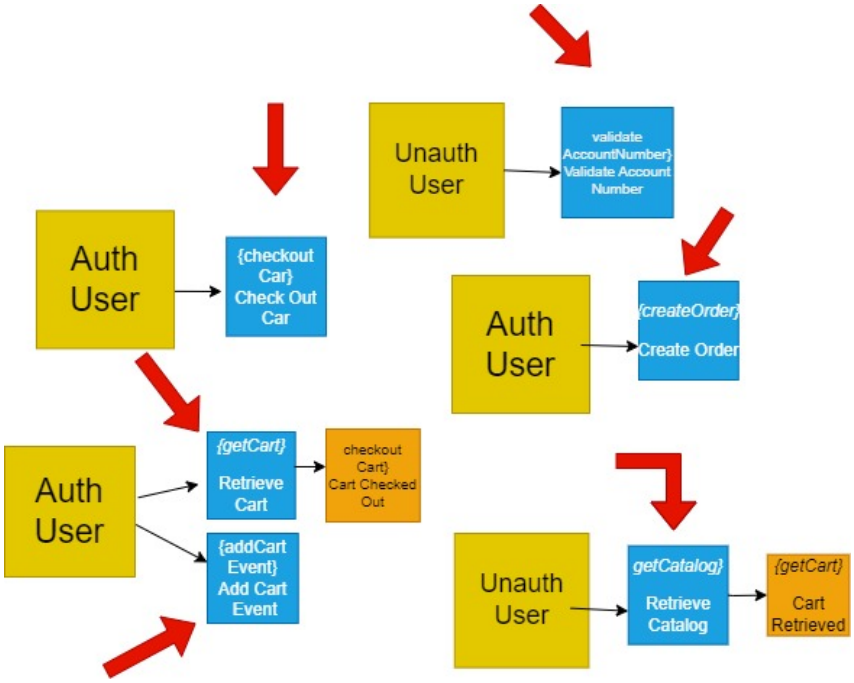
5.2.1 *EventStorming* para Análise de Superfícies de Ataque

A aplicação da técnica de *EventStorming* na elaboração do mapeamento da superfície de ataque oferece diversas vantagens que contribuem para a segurança dos sistemas desde a fase inicial de definição dos requisitos, muito antes do início efetivo do desenvolvimento. Nesse contexto, é importante destacar que, tradicionalmente, os Diagramas de Fluxo de Dados (*DFDs*) têm sido a ferramenta mais utilizada para identificar a superfície de ataque, conforme demonstrado em pesquisas anteriores (THEISEN et al., 2017; LIU et al., 2020).

Contudo, o *EventStorming* se sobressai por valorizar a comunicação entre todos os envolvidos, permitindo que os *stakeholders* analisem o sistema de forma coletiva. Essa abordagem possibilita uma identificação ágil e iterativa dos possíveis pontos vulneráveis do sistema, conforme ilustrado na Figura 13. Assim, com a *Big Picture* (Figura 16, disponível no Apêndice 7.3) elaborada de maneira colaborativa, foi possível examinar os comandos,

eventos e seus respectivos usuários, identificando um total de 17 ações, das quais 13 estão expostas na internet por meio do acesso à *API*. Esses resultados demonstram que a *Big Picture* pode constituir um artefato valioso na análise da superfície de ataque.

Figura 13 – Primeiro Cenário: Mapeamento da Superfície de Ataque com *EventStorming*



Ao contrário da abordagem representada pelo *Interactive Application Security Testing (IAST)* (ZHANG, 2023), que requer a existência de um sistema implementado, a análise da superfície de ataques, realizada na fase de design de software, representa uma prática que ostenta uma série de méritos para a segurança e eficácia de sistemas. Esta prática permite a identificação dos elementos suscetíveis a ataques antes de sua efetiva implementação, configurando, assim, uma abordagem proativa na mitigação de vulnerabilidades e riscos de segurança desde os estágios iniciais do desenvolvimento. O trabalho de Bradbury et al. (BRADBURY et al., 2020) traz uma abordagem que usa um modelo de referência, apesar de eficiente, tal modelo está aplicado em um contexto específico. Em geral, os trabalhos adotam *UML* (HERRMANN, 2001; JÜRJENS; SCHRECK; BARTMANN, 2008), criando um formalismo para análise de superfície de ataques. Como diferencial, este trabalho adotou o *EventStorming*, que traz um formalismo menor que a *UML*, porém, permite a análise de superfície de ataques de forma integrada aos modelos ágeis.

Vale ressaltar que, embora seja amplamente aconselhável a aplicação do *EventStorming* antes da etapa de implementação, este modelo também pode ser empregado como uma ferramenta de reengenharia, conforme ilustrado no relato de experiência em questão. Neste cenário, o método é aplicado a um produto já existente, no qual se realiza um *workshop* para identificar os atores, comandos, eventos, políticas e outros recursos envolvidos.

Posteriormente, a criação de uma *Big Picture* da visão atual do sistema viabiliza a análise da superfície de ataques. Como resultado, com base nas conclusões dessa análise, uma nova *Big Picture* é gerada, contendo recomendações para a refatoração do sistema.

Este modelo, embora envolva um esforço adicional, proporciona aos *stakeholders* a oportunidade ágil de avaliar o estado atual do sistema. Isso permite a tomada de decisões embasadas no impacto das exposições identificadas na superfície de ataque e a seleção das alterações a serem efetuadas, promovendo, uma abordagem fundamentada na gestão proativa da segurança do software.

5.2.2 Análise Comparativa entre *EventStorming* e *DFDs*

A seleção da metodologia para a análise da superfície de ataque em arquiteturas de microsserviços desempenha um papel crucial na identificação e mitigação de vulnerabilidades. Diferentes abordagens apresentam características distintas quanto ao foco, nível de abstração e aplicabilidade, influenciando diretamente a eficácia da detecção de ameaças. Neste contexto, o presente estudo propõe a utilização do *EventStorming* como técnica para modelagem da superfície de ataque e realiza uma comparação teórica com os Diagramas de Fluxo de Dados *(DFDs)*, que representam uma abordagem consolidada na análise de ameaças.

Tabela 11 – Comparação entre as Técnicas: *EventStorming* e Diagramas de Fluxo de Dados *(DFDs)*

Critério	*EventStorming*	**Diagramas de Fluxo de Dados *(DFDs)***
Foco	Eventos e interações	Fluxo de dados e armazenamento.
Abordagem	Exploratória e colaborativa	Estruturada e analítica.
Nível de Abstração	Alto nível, modelagem dinâmica	Diferentes níveis, modelagem estática.
Facilidade de Uso	Alta, devido à colaboração visual	Moderada, requer conhecimento técnico
Identificação de Vulnerabilidades	Situacional e orientada a eventos relevantes	Baseada em fluxo de dados e controle de acesso.
Ferramentas Associadas	*Post-its*	*Microsoft Threat Modeling Tool*.
Eficiência na Segurança	Alta (identificação de superfícies de ataque emergentes)	Média (bom para modelagem tradicional).
Adaptação a Microsserviços	Alta (eventos são naturais em microsserviços)	Baixa a Média (necessita adaptações).

O *EventStorming* caracteriza-se como uma técnica exploratória e colaborativa, voltada para a compreensão dos eventos e interações dentro de um sistema. Sua modelagem ocorre em um nível abstrato e dinâmico, proporcionando uma visão holística dos processos e favorecendo a identificação de ameaças emergentes. Um dos principais atributos dessa abordagem é sua acessibilidade, uma vez que sua execução se baseia em dinâmicas visuais e no uso de ferramentas simples, como *post-its*, facilitando o mapeamento dos fluxos de eventos e a discussão de possíveis vulnerabilidades. Além disso, sua compatibilidade com arquiteturas de microsserviços deve-se ao fato de que eventos são elementos centrais desse modelo, tornando a análise mais intuitiva e eficaz. Assim, o *EventStorming* mostra-se especialmente relevante para a identificação de superfícies de ataque dinâmicas e vulnerabilidades situacionais.

Por outro lado, os Diagramas de Fluxo de Dados *(DFDs)* adotam uma abordagem mais estruturada e analítica, concentrando-se no fluxo de dados e nos mecanismos de armazenamento. Essa técnica permite representar o sistema em diferentes níveis de abstração, possibilitando tanto uma visão macro quanto uma análise detalhada de processos específicos. No entanto, sua aplicação exige um conhecimento técnico mais aprofundado, o que pode limitar a participação colaborativa durante a modelagem. A detecção de vulnerabilidades em *DFDs* está essencialmente associada ao controle de acesso e ao monitoramento do fluxo de dados, sendo uma metodologia amplamente utilizada na modelagem tradicional de ameaças. Ferramentas como a *Microsoft Threat Modeling Tool* são frequentemente empregadas para esse propósito. Entretanto, sua adaptação a arquiteturas de microsserviços apresenta limitações, uma vez que esses sistemas são altamente distribuídos e orientados a eventos, dificultando a representação precisa dos fluxos de informação.

No que tange à eficiência na segurança, o *EventStorming*, por sua abordagem dinâmica e colaborativa, destaca-se na identificação de ameaças emergentes e no mapeamento contextual de eventos críticos, enquanto os *DFDs* demonstram maior adequação à modelagem tradicional, sendo mais eficazes em ambientes centralizados e menos dinâmicos. Dessa forma, a escolha entre essas metodologias deve considerar fatores como o grau de colaboração desejado, a complexidade da arquitetura analisada e o perfil das ameaças a serem identificadas.

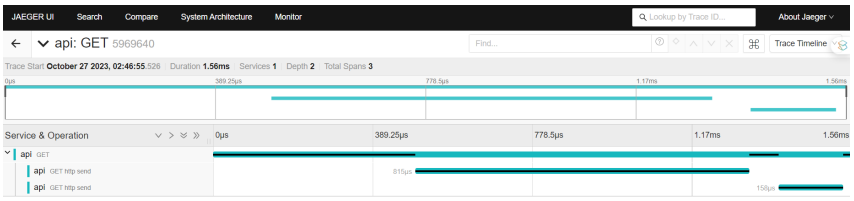
5.2.3 Uso de *Behavior-Driven Development (BDD)* para Requisitos Não Funcionais

Neste estudo, o *Behavior-Driven Development (BDD)* foi aplicado para a elicitação de requisitos não-funcionais, focando na característica de eficiência de desempenho da Norma *ISO/IEC/IEEE 25010* e suas subcaracterísticas (comportamento em relação ao tempo, utilização de recursos e capacidade). Cada característica foi associada a histórias de usuário

e cenários de aceitação, conforme detalhado na Subseção de execução 4.5.1, o que permitiu orientar os testes de usuário e vincular os requisitos à linguagem *gherkin* do *BDD*.

Para a automação dos testes e monitorização do desempenho, empregou-se um conjunto de ferramentas. A comunicação entre os serviços foi analisada com o uso do *Jaeger*, possibilitando a visualização do desempenho e das interações entre microsserviços, conforme ilustrado na Figura 14.

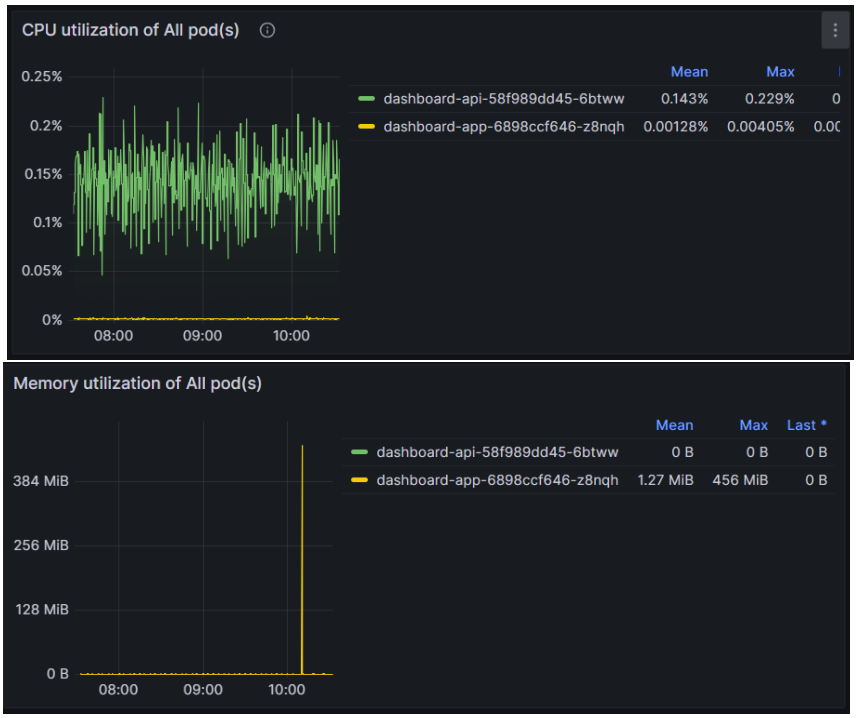
Figura 14 – Comunicação entre Microsserviços com o uso do *Jaeger*



O *Jaeger* apresentou a comunicação entre os serviços, demonstrando seu desempenho e a forma como cada microsserviço interage, permitindo identificar potenciais problemas internos entre eles.

A Figura 15 ilustra a análise dos *containers* dos serviços em execução realizada pelo Grafana, que permite compreender o consumo de memória e processador durante os testes. Essa integração possibilitou o cruzamento de informações da perspectiva do usuário (com *Locust*), do serviço (com *Jaeger*) e do servidor (com *Grafana*), proporcionando uma visão 360° do desempenho do sistema.

Figura 15 – Avaliação dos Componentes de Infraestrutura (CPU e Memória) com Grafana



A avaliação dos componentes de infraestrutura, acompanhada pelo Grafana, proporcionou uma visão abrangente e *end-to-end* do desempenho do sistema.

5.3 Experimento com Modelos de Grande Linguagem *(LLMs)* para Segurança

Esta seção detalha os resultados do experimento focado na aplicação de Modelos de Grande Linguagem *(LLMs)* no contexto da segurança, especificamente explorando a similaridade de respostas entre diferentes *LLMs* e a aplicabilidade do *Behavior-Driven Development (BDD)* com *LLMs* para requisitos não-funcionais de segurança.

Q1 - Similaridade das respostas entre *ChatGPT e Gemini*

A fim de mensurar a similaridade entre as respostas geradas pelo *ChatGPT* e pelo *Gemini*, foi aplicado o teste estatístico não paramétrico de *Kruskal-Wallis*, adequado para a comparação de distribuições independentes. As hipóteses formuladas para a condução dessa análise são apresentadas a seguir.

- Métrica: Coeficiente de similaridade.
- Hipótese Alternativa (H1): Os *LLMs* não geram similaridade entre si.
- Hipótese Nula (HN1): Os *LLMs* geram similaridade entre si.

Os resultados do teste de Kruskal-Wallis para cada atributo de segurança são detalhados a seguir:

Confidencialidade Estatística: 6,8181 Valor de p: 0,0090 Os resultados indicaram uma estatística de *Kruskal-Wallis* igual a 6,8181, com um valor de p de 0,0090. Esses resultados evidenciam diferenças significativas entre os *LLMs*. O valor de p, significativamente inferior a 0,05, sugere a existência de uma diferença estatisticamente significativa na similaridade das respostas entre *ChatGPT* e *Gemini*.

Integridade Estatística: 19,1731 Valor de p: 0,0017 Os resultados indicaram uma estatística de *Kruskal-Wallis* de 19,1731 e um valor de p de 0,0017. Esses testes evidenciam diferenças significativas entre os *LLMs*. O valor de p, significativamente inferior a 0,05, sugere que existe uma diferença estatisticamente significativa na similaridade das respostas entre *ChatGPT* e *Gemini*.

Não-repúdio Estatística: 9,81 Valor de p: 0,0074 Os resultados indicaram uma estatística de *Kruskal-Wallis* de 9,81 e um valor de p de 0,0074. Os testes apresentaram evidências de diferenças entre os *LLMs*. O valor de p, significativamente inferior a 0,05, sugere a existência de uma diferença estatisticamente significativa na similaridade das respostas entre *ChatGPT* e *Gemini*.

Responsabilização Estatística: 5,33 Valor de p: 0,0209 Os resultados indicaram uma estatística de *Kruskal-Wallis* de 5,33 e um valor de p de 0,0209. Os testes evidenciaram diferenças significativas entre os *LLMs*. O valor de p, inferior a 0,05, sugere a existência de uma diferença estatisticamente significativa na similaridade das respostas entre *ChatGPT* e *Gemini*.

Autenticidade Estatística: 5,33 Valor de p: 0,0209 Os resultados indicaram uma estatística de *Kruskal-Wallis* de 5,33 e um valor de p de 0,0209. Os testes evidenciaram diferenças significativas entre os *LLMs*. O valor de p, inferior a 0,05, sugere a existência de uma diferença estatisticamente significativa na similaridade das respostas entre *ChatGPT* e *Gemini*.

Resistência Estatística: 5,33 Valor de p: 0,021 Os resultados indicaram uma estatística de Kruskal-Wallis de 5,33 e um valor de p de 0.021. Os testes apresentaram evidências de diferenças entre os *LLMs*. O valor de p, significativamente inferior a 0,05, sugere a existência de uma diferença estatisticamente significativa na similaridade das respostas entre *ChatGPT* e *Gemini*.

Os testes mostraram evidências de que há diferenças entre os grupos, resultando na confirmação da Hipótese Alternativa (H1).

Q2 - Como o *BDD* pode ser usado para garantir a qualidade relacionada aos requisitos não funcionais de segurança por meio de um *LLMs*?

A utilização do *Behavior-Driven Development (BDD)* pode contribuir eficazmente para a elicitação de requisitos não funcionais, especialmente quando associada à clareza e concisão na elaboração dos *prompts*. Por meio do padrão *Gherkin*, o *BDD* favorece a formulação de frases curtas, objetivas e de fácil interpretação, o que potencializa a precisão dos resultados obtidos.

Modelos de linguagem de grande escala (*LLMs*), como o *ChatGPT* e *Gemini*, ampliam essas possibilidades ao oferecer vantagens significativas, tais como:

- **Criação e validação de requisitos**: Modelos de linguagem como o *ChatGPT* e *Gemini* contribuem significativamente para a formulação e validação de requisitos não funcionais especialmente os relacionados à segurança de maneira clara, objetiva e tecnicamente consistente.
- **Coleta de requisitos**: Na fase de levantamento, esses modelos demonstram a capacidade de interpretar dados complexos inseridos nos *prompts*, convertendo-os em especificações precisas. Tal abordagem possibilita a construção de cenários de teste alinhados à norma *ISO/IEC 25010:2023*, especialmente quando integrados a metodologias como o *Behavior-Driven Development (BDD)*.

- **Automação de documentação**: Os *LLMs* ampliam a capacidade de automação na elaboração de artefatos, facilitando a geração estruturada e padronizada de documentação técnica e cenários de teste.
- **Produtividade e consistência**: A automação proporcionada por esses modelos reduz significativamente o esforço manual e o tempo demandado para a elaboração de documentos. Além disso, a produção textual consistente contribui para a padronização e a qualidade dos requisitos especificados.

A partir da elaboração de *prompts* tecnicamente adequados, verifica-se a viabilidade de automatizar os testes dos requisitos não funcionais abordados neste estudo por meio da abordagem *Behavior-Driven Development (BDD)*, assegurando maior assertividade na verificação do comportamento esperado desses requisitos.

6
Discussão

Este capítulo apresenta a discussão do trabalho desenvolvido, analisando-o criticamente e estabelecendo uma conexão intrínseca com os achados detalhados no capítulo de análise de resultados.

6.1 Mapeamento Sistemático de Literatura

No contexto da análise da superfície de ataque em sistemas com arquitetura de microsserviços, o mapeamento sistemático de literatura permitiu a identificação e extração de informações cruciais. Este processo visou subsidiar a elaboração de estratégias de segurança mais eficazes, possibilitando uma tomada de decisão assertiva. A pesquisa focou na identificação e avaliação de potenciais vulnerabilidades, considerando a natureza distribuída e a comunicação inerente aos microsserviços, com o objetivo de reduzir os vetores de ataque e fortalecer a proteção do sistema.

6.2 Análise da Superfície de Ataque com *EventStorming*

A escolha da metodologia para a análise da superfície de ataque em arquiteturas de microsserviços desempenha um papel fundamental na identificação e mitigação de vulnerabilidades. Diferentes abordagens apresentam características específicas em termos de foco, nível de abstração e aplicabilidade, influenciando diretamente a eficácia na detecção de ameaças. Entre essas abordagens, destacam-se o *EventStorming* e os Diagramas de Fluxo de Dados (*DFDs*), que oferecem perspectivas complementares na modelagem de riscos de segurança.

O *EventStorming* caracteriza-se como uma técnica exploratória e colaborativa, voltada à compreensão dos eventos e interações dentro de um sistema. Sua modelagem ocorre em um

nível abstrato e dinâmico, proporcionando uma visão holística dos processos e favorecendo a identificação de ameaças emergentes. Um de seus principais atributos é a acessibilidade, uma vez que se baseia em dinâmicas visuais e no uso de ferramentas simples, como *post-its*, facilitando o mapeamento dos fluxos de eventos e a discussão sobre possíveis vulnerabilidades. Além disso, sua compatibilidade com arquiteturas de microsserviços decorre do fato de os eventos constituírem elementos centrais nesse modelo, tornando a análise mais intuitiva e eficaz. Dessa forma, o *EventStorming* revela-se especialmente relevante para a identificação de superfícies de ataque dinâmicas e vulnerabilidades situacionais.

Em contraste, os *DFDs* adotam uma abordagem mais estruturada e analítica, com foco no fluxo de dados e nos mecanismos de armazenamento. Essa técnica permite representar o sistema em diferentes níveis de abstração, viabilizando tanto uma visão macro quanto uma análise detalhada de processos específicos. No entanto, sua aplicação requer um conhecimento técnico mais aprofundado, o que pode restringir a participação colaborativa durante a modelagem. A detecção de vulnerabilidades por meio dos *DFDs* está essencialmente associada ao controle de acesso e ao monitoramento dos fluxos de dados, sendo uma metodologia consolidada na modelagem tradicional de ameaças. Ferramentas como a *Microsoft Threat Modeling Tool* são frequentemente utilizadas para esse propósito. Contudo, sua adaptação a arquiteturas de microsserviços é limitada, uma vez que tais sistemas são altamente distribuídos e orientados a eventos, o que pode dificultar a representação precisa dos fluxos de informação.

No que se refere à eficiência em segurança, o *EventStorming* destaca-se pela identificação de ameaças emergentes e pelo mapeamento contextual de eventos críticos, enquanto os *DFDs* demonstram maior adequação à modelagem tradicional, sendo mais eficazes em ambientes centralizados e menos dinâmicos. Assim, a escolha entre essas metodologias deve considerar fatores como o grau de colaboração desejado, a complexidade da arquitetura em análise e o perfil das ameaças a serem identificadas.

6.2.1 Comparação das Técnicas de Análise da Superfície de Ataque

Tabela 12 – Comparação das Técnicas de Análise da Superfície de Ataque

Critério	**EventStorming**	**Diagramas de Fluxo de Dados (DFD)**
Foco	Eventos e interações	Fluxo de dados e armazenamento
Abordagem	Exploratória e colaborativa	Estruturada e analítica
Nível de abstração	Alto nível, modelagem dinâmica	Diferentes níveis, modelagem estática
Facilidade de uso	Alta, devido à colaboração visual	Moderada, requer conhecimento técnico
Identificação de vulnerabilidades	Contextual, baseada em eventos críticos	Baseada no fluxo de dados e controle de acesso
Ferramentas associadas	*Post-its*	*Microsoft Threat Modeling Tool*
Eficiência na segurança	Alta	Média
Adaptação a microsserviços	Alta	Baixa a média (necessita adaptações)

6.3 Uso de *Behavior-Driven Development (BDD)* para Requisitos Não Funcionais

A discussão dos resultados obtidos revela a eficácia da abordagem proposta para a elicitação e validação de requisitos não-funcionais. Os *frameworks* do *BDD*, como Cucumber e JBehave, embora inicialmente concebidos para unidades funcionais, demonstraram que, com o uso de ferramentas complementares *(Locust, Jaeger, Grafana)*, é possível mitigar substancialmente os desafios de sua aplicação em cenários de requisitos não-funcionais. A integração dessas ferramentas permitiu uma automação robusta dos testes e uma monitorização detalhada do desempenho.

Conforme mencionado por Haoues *et al.* (HAOUES et al., 2017), a manutenção da qualidade do software, em conformidade com a Norma *ISO/IEC/IEEE 25010*, exige um foco contínuo no aspecto de qualidade ao longo de todo o ciclo de vida do software. Nesse contexto, o *BDD* demonstrou resultados positivos desde a elicitação dos requisitos, caracterizada pela objetividade na interação entre os membros da equipe, até a automação dos testes, que permitiu atender aos critérios de aceite propostos. A linguagem *Gherkin* do *BDD* revelou-se um valioso auxílio no direcionamento para o comportamento esperado do sistema, garantindo que as histórias de usuário fossem descritas com um propósito claro.

A respeito da Revisão Sistemática de Literatura focada em requisitos não-funcionais, realizada por Olsson, Sentilles e Papatheocharous (OLSSON; SENTILLES; PAPATHEOCHAROUS, 2022), os autores destacaram a necessidade de estudos que validassem as pesquisas acadêmicas com a realidade da indústria. Dessa forma, esta pesquisa contribui significativamente para o entendimento da elicitação de requisitos não-funcionais com o *BDD*, por ter sido aplicada em um cenário prático e real, fornecendo validação empírica.

Adicionalmente, conforme North (NORTH, 2006) preconizou, o *BDD* foi criado para mitigar os problemas do Test-Driven Development (*TDD*), focando no comportamento das *releases*. Os benefícios do *BDD*, como a melhoria na comunicação e legibilidade, foram claramente percebidos nesta pesquisa durante a elicitação dos requisitos não-funcionais. A participação ativa dos envolvidos no desenvolvimento das *releases*, conforme descrito na Subseção 4.5.1, reforça a eficácia dessa abordagem colaborativa.

6.4 Experimento com Modelos de Grande Linguagem (*LLMs*) para Segurança

O experimento descrito neste artigo estabeleceu uma conexão entre a geração automatizada de código de teste e a colaboração entre humanos e algoritmos de aprendizado de máquina. Demonstrou-se que, por meio de *prompts* estruturados com objetivos claros, as inteligências artificiais (*IAs*) podem efetivamente auxiliar na produção de critérios de aceitação de alta qualidade. O modelo de *prompt* desenvolvido para este experimento será detalhado nas seções subsequentes.

Rajbhoj et al. (RAJBHOJ et al., 2024) destacaram a necessidade de avançar os estudos sobre *IAs* na fase de elicitação de requisitos dentro do processo de desenvolvimento de software. Neste experimento, conseguimos apresentar resultados aceitáveis por meio dos *prompts* utilizados. Além disso, o estudo de Kasauli et al. (KASAULI et al., 2021) apontou a necessidade de uma abordagem sólida para a elicitação de requisitos, enquanto o trabalho de Jarzkebowicz e Weichbroth (JARZĘBOWICZ; WEICHBROTH, 2021) revelou que a elicitação de requisitos não-funcionais varia entre diferentes empresas. Nesse contexto, os *prompts* criados neste experimento podem ser uma alternativa para ambos os problemas, oferecendo uma solução padronizada e eficaz.

Santos et al. (SANTOS et al., 2024b) realizaram um estudo de caso sobre a eficácia do *BDD* na elicitação de requisitos não-funcionais. O experimento apresentado neste artigo demonstrou que a eficácia do *BDD* se estende ao uso de Modelos de Linguagem de Grande Escala (*LLMs*), especialmente quando se trata de requisitos não-funcionais, automatizando o processo e gerando economia de tempo. Adicionalmente, este estudo contribui para o trabalho de Olsson, Sentilles e Papatheocharous (OLSSON; SENTILLES;

PAPATHEOCHAROUS, 2022), pois se trata de um estudo empírico conduzido em uma situação real.

Ao criar o *BDD*, Dan North buscou mitigar os problemas relacionados a testes oriundos do Desenvolvimento Orientado por Testes (*TDD*) (NORTH, 2006). Utilizando a linguagem *Gherkin*, o *BDD* é adotado de forma sistemática e concisa, otimizando o tempo gasto em partes do processo de desenvolvimento de software, como a elicitação de requisitos. A combinação do uso do *BDD* com *LLMs* oferece uma abordagem prática que as empresas podem adotar para auxiliar as equipes em atividades rotineiras, liberando mais tempo para outras tarefas críticas de desenvolvimento de software.

6.5 Ameaças à Validade

Entende-se por ameaça à validade os aspectos mitigados durante a pesquisa com o objetivo de garantir a confiabilidade do estudo (RUNESON; HÖST, 2009). Para a identificação e tratamento dessas ameaças, foram utilizadas as descrições de Zhou *et al*. (ZHOU et al., 2016) sobre os tipos de validade.

6.5.1 Validade de Construção

Para o Mapeamento Sistemático da Literatura, minimizou-se a ameaça de utilizar termos de pesquisa inadequados ou incompletos na busca automatizada mediante a realização de um pré-teste da sequência de busca antes da execução do protocolo de pesquisa. Essa etapa garantiu que os resultados obtidos fossem relevantes e alinhados aos objetivos do estudo, permitindo ajustar os termos e evitar potenciais falhas que poderiam comprometer a abrangência e a precisão dos dados coletados, aumentando a confiabilidade.

Para o Experimento com *LLMs* e *BDD*, a compreensão aprofundada da Norma *ISO/IEC 25010:2023*, do *framework BDD* e do funcionamento dos *LLMs* foi fundamental. Essa base de conhecimento, detalhada na Seção 2, possibilitou analisar a conformidade da elicitação de requisitos não-funcionais, via *LLM* e *BDD*, com os requisitos de segurança definidos na Norma *ISO/IEC 25010:2023*.

6.5.2 Validade Interna

Para o Mapeamento Sistemático da Literatura, minimizou-se o viés na seleção de estudos e na extração de dados por meio de um processo colaborativo, onde dois pesquisadores realizaram essas tarefas de forma independente. Após a fase inicial, as decisões sobre a adequação de cada estudo foram revisadas e comparadas de forma cruzada, garantindo uma análise mais robusta e imparcial. Divergências foram discutidas e resolvidas em conjunto, assegurando maior rigor metodológico e confiabilidade na inclusão dos estudos selecionados.

Para o Experimento com *LLMs* e *BDD*, a pesquisa foi conduzida com a participação de cinco colaboradores: três foram responsáveis pela coleta de informações sobre a Norma *ISO/IEC 25010:2023*, *BDD* e *LLMs*; dois realizaram o experimento; e, por fim, o último revisou e refinou as informações ao longo de todo o processo de escrita do artigo.

6.5.3 Validade Externa

Para o Mapeamento Sistemático da Literatura, assegurou-se a generalização dos resultados com a utilização da ferramenta *Parsifal* e uma seleção criteriosa de estudos que representassem uma ampla variedade de ambientes e arquiteturas de microsserviços. Os estudos incluídos abordaram diferentes superfícies de ataque, bem como ambientes com diversos graus de criticidade e complexidade. A metodologia PICO foi aplicada para estruturar essa análise, garantindo o foco em populações relevantes, intervenções que contemplassem diversas abordagens de segurança, e comparações com alternativas eficazes, reforçando a aplicabilidade das conclusões a um cenário mais amplo.

Para o Experimento com *LLMs* e *BDD*, a validade deste estudo foi aprovada por teste estatístico, o que permite a replicação do experimento a partir das etapas metodológicas expressas na Seção 5.3. O teste estatístico realizado conferiu confiabilidade aos resultados do estudo.

6.5.4 Validade de Conclusão

Para o Mapeamento Sistemático da Literatura, garantiu-se resultados replicáveis com o estabelecimento de um protocolo rigoroso validado por meio de revisões cruzadas. Essa abordagem permitiu que os pesquisadores trabalhassem de forma independente, conduzindo descobertas individuais que, posteriormente, foram discutidas e validadas conjuntamente, assegurando a consistência e a precisão das conclusões.

Para o Experimento com *LLMs* e *BDD*, os pesquisadores não apresentaram objeções quanto às etapas metodológicas utilizadas e aos resultados obtidos. Este estudo, portanto, apresenta resultados verificados e validados.

6.6 Limitações

O *EventStorming* tradicional pode não contemplar todos os aspectos essenciais relacionados à segurança, como técnicas de ataque avançadas ou vulnerabilidades específicas. Além disso, não considera as contramedidas de segurança nem o impacto que estas podem exercer sobre a superfície de ataque.

Dessa forma, torna-se necessário adaptar o método para incorporar essa perspectiva, garantindo assim sua eficácia. A inclusão de eventos e fluxos de dados associados à

segurança pode tornar o modelo mais complexo e difícil de compreender e gerenciar, especialmente se não for realizada de forma clara. Portanto, embora o *EventStorming* seja uma ferramenta valiosa para modelar a superfície de ataque, é fundamental ajustá-lo adequadamente e considerar suas limitações para obter uma visão abrangente e eficaz da segurança do sistema.

A principal limitação ao adaptar o *EventStorming* para a análise de requisitos não funcionais, como a segurança, reside na necessidade da participação de um especialista em segurança durante o *workshop*. A presença desse especialista foi decisiva para a identificação de vulnerabilidades nos eventos do domínio. Para superar essa restrição, optou-se por convidar um especialista em microsserviços, o que garantiu uma análise eficiente dos pontos críticos e uma consideração adequada dos aspectos relacionados à segurança.

7

Conclusões

Este trabalho abordou o desafio crítico da **segurança da informação em arquiteturas baseadas em microsserviços**, propondo e avaliando um modelo inovador para a análise da superfície de ataque e requisitos não funcionais. Utilizando a metodologia *Design Science Research*, que incluiu um Mapeamento Sistemático de Literatura para aquisição de conhecimento e validação por meio de um relato de experiência e um estudo de caso, foi possível desenvolver uma abordagem que aprimora significativamente as práticas de segurança em sistemas distribuídos.

Os resultados obtidos demonstram a **eficácia inegável da proposta metodológica no fortalecimento da segurança da informação**. A aplicação da técnica de **EventStorming** revelou-se uma ferramenta poderosa para a **identificação e análise proativa da superfície de ataque**. Através dela, foi possível realizar um mapeamento minucioso de vulnerabilidades e a *pinpointing* de pontos críticos de exposição que, de outra forma, poderiam passar despercebidos. Essa capacidade de visibilidade aprimorada e a promoção da colaboração entre equipes são fundamentais para uma postura de segurança mais robusta, permitindo uma compreensão mais profunda das complexas interações e dependências que são inerentes aos microsserviços e que frequentemente representam vetores de ataque.

Adicionalmente, a integração da abordagem **Behavior-Driven Development (BDD)** no estudo de caso foi crucial para a **verificação sistemática de requisitos não funcionais, com um enfoque particular na segurança** e no desempenho. Os benefícios evidentes na automação de testes e na rastreabilidade dos requisitos não apenas aumentaram a confiabilidade geral do sistema, mas também garantiram que as preocupações de segurança fossem abordadas desde as fases iniciais do desenvolvimento, e de forma verificável. Este aspecto é vital para a criação de software mais seguro, pois permite que falhas de segurança sejam identificadas e mitigadas precocemente.

Em síntese, o modelo proposto representa uma **contribuição substancial para a enge-**

nharia de software orientada à segurança em ambientes distribuídos. Ao fornecer um arcabouço prático e escalável para a análise da superfície de ataque e a avaliação de requisitos de segurança, esta pesquisa não apenas avança o conhecimento teórico no campo, mas também oferece subsídios concretos para que organizações possam construir e manter sistemas de microsserviços com um nível significativamente maior de resiliência e proteção contra ameaças cibernéticas.

7.1 Trabalhos Futuros

Nossas direções futuras de pesquisa serão construídas sobre a base teórica e metodológica estabelecida por trabalhos prévios e o estudo aqui apresentado.

Fundamentação para Pesquisas Futuras

A base para a evolução de nosso estudo inclui a análise de dois artigos cruciais. Primeiramente, *"An empirical study of security practices for microservices systems"* serviu como um alicerce para identificar e extrair um catálogo de 28 práticas de segurança de microsserviços, fundamentais para um estudo de caso focado em boas práticas de desenvolvimento integradas ao *Behavior-Driven Development (BDD)*. Em segundo lugar, o trabalho *"Building Secure Environments for Microservices"* ofereceu uma metodologia robusta para detecção de configurações de sistemas vulneráveis em ambientes de microsserviços, avaliação de riscos cibernéticos e reconfiguração para maior segurança. Esta metodologia justifica o uso de testes de segurança baseados em padrões como : *Common Criteria, The Open Group Architecture Framework (TOGAF), Security Assurance Maturity Model (SAMM), Building Security In Maturity Model (BSIMM), Application Security Verification Standard (ASVS), OWASP, SAFECode, e normas como PCI, NIST e ISO/IEC*.

Direções de Pesquisa Futura

Como trabalho futuro direto, pretendemos explorar a aplicação de Modelos de Linguagem de Grande Escala *(LLMs)* para a geração automática de cenários de ataque. Nosso objetivo é integrar esses cenários diretamente ao processo de *EventStorming* de segurança, além de investigar a aplicabilidade de *LLMs* em diversos domínios e contextos da segurança cibernética.

7.2 Produções

Durante o período do mestrado, foram desenvolvidas as seguintes produções acadêmicas, que refletem os avanços e resultados alcançados na pesquisa:

Artigo Publicado: "*Using Behavior-Driven Development (BDD) for Non-functional Requirements*" no *Journal of Advance Research in Computer Science Engineering (ISSN*

2456-3552). Este trabalho explora a aplicação do BDD para a elicitação e validação de requisitos não-funcionais.

Artigo Submetido *(*Webist): "*Attack Surface Analysis in Systems with Microservice Architecture: A Systematic Mapping*". Esta submissão apresenta os resultados de um mapeamento sistemático da literatura sobre a análise da superfície de ataque em arquiteturas de microsserviços.

Relato de Experiência Submetido *(*Webist): "*Using Event Storming in attack surface analysis: a report of experience for a microservices architecture system*". Este relato detalha uma experiência prática da aplicação do *EventStorming* para a análise de superfície de ataque em um sistema de microsserviços.

Trabalho em Desenvolvimento: "*An experiment with focus on security through Large-Language Models using Behavior-Driven Development*". Este estudo em andamento investiga o uso de Modelos de Linguagem de Grande Escala *(LLMs)* em conjunto com o *BDD* para aprimorar a segurança de sistemas.

Referências

ADAMOS, K. et al. Enhancing attack resilience of cyber-physical systems through state dependency graph models. Int. J. Inf. Secur., Springer-Verlag, Berlin, Heidelberg, v. 23, n. 1, p. 187–198, jul. 2023. ISSN 1615-5262. Disponível em: <https://doi.org/10.1007/s10207-023-00731-w>. Citado na página 97.

AKEN, J. van; ROMME, G. A design science approach to evidence-based management. In: ____. [S.l.: s.n.], 2012. p. 43–57. ISBN 0199763984. Citado na página 32.

ALABBAD, M.; MHASKAR, N.; KHEDRI, R. Two formal design solutions for the generalization of network segmentation. Journal of Network and Computer Applications, v. 222, p. 103763, 2024. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804523001820>. Citado na página 102.

ALJAWAWDEH, H.; ALJAIDI, M.; MAGHRABI, L. Towards serverless & microservices architecture: Strategies, challenges, and insights into technology. In: ____. Artificial Intelligence and Economic Sustainability in the Era of Industrial Revolution 5.0. Cham: Springer Nature Switzerland, 2024. p. 447–458. ISBN 978-3-031-56586-1. Disponível em: <https://doi.org/10.1007/978-3-031-56586-1_33>. Citado na página 102.

ALJAWAWDEH, H.; SABRI, M. O.; MAGHRABI, L. Toward serverless and microservices architecture: Literature, methods, and best practices. In: ____. [S.l.: s.n.], 2023. p. 573–584. ISBN 978-3-031-43299-6. Citado na página 102.

ALKADI, O.; MOUSTAFA, N.; TURNBULL, B. A review of intrusion detection and blockchain applications in the cloud: Approaches, challenges and solutions. IEEE Access, v. 8, p. 104893–104917, 2020. Citado 2 vezes nas páginas 56 e 95.

ANTUNES, J. F. . N. Evaluating intrusion detection for microservice applications: Benchmark, dataset, and case studies. Journal of Systems and Software, v. 216, p. 112142, 2024. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121224001870>. Citado 2 vezes nas páginas 56 e 97.

ARAUJO, I.; ANTUNES, N.; VIEIRA, M. Evaluation of machine learning for intrusion detection in microservice applications. In: Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing. New York, NY, USA: Association for Computing Machinery, 2023. (LADC '23), p. 126–135. ISBN 9798400708442. Disponível em: <https://doi.org/10.1145/3615366.3615375>. Citado na página 97.

ARAUJO, I.; ANTUNES, N.; VIEIRA, M. Intrusion detection and tolerance for microservice applications. In: Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing. New York, NY, USA: Association for Computing Machinery, 2023. (LADC '23), p. 176–181. ISBN 9798400708442. Disponível em: <https://doi.org/10.1145/3615366.3622794>. Citado na página 98.

BANDARA, E. et al. Let'strace — blockchain, federated learning and tuf/in-toto enabled cyber supply chain provenance platform. In: . [S.l.: s.n.], 2021. p. 470–476. Citado na página 98.

BANGUI, H.; ROSSI, B.; BUHNOVA, B. A conceptual antifragile microservice framework for reshaping critical infrastructures. In: 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME). [S.l.: s.n.], 2022. p. 364–368. Citado 2 vezes nas páginas 56 e 95.

BEEKMAN, J. G.; PORTER, D. E. Challenges for scaling applications across enclaves. In: Proceedings of the 2nd Workshop on System Software for Trusted Execution. New York, NY, USA: Association for Computing Machinery, 2017. (SysTEX'17). ISBN 9781450350976. Disponível em: <https://doi.org/10.1145/3152701.3152710>. Citado na página 96.

BEHRAD, S. et al. Impacts of service decomposition models on security attributes: A case study with 5g network repository function. In: 2021 IEEE 7th International Conference on Network Softwarization (NetSoft). [S.l.: s.n.], 2021. p. 470–476. Citado na página 98.

BéLAIR, M.; LANIEPCE, S.; MENAUD, J.-M. Snappy: programmable kernel-level policies for containers. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing. New York, NY, USA: Association for Computing Machinery, 2021. (SAC '21), p. 1636–1645. ISBN 9781450381048. Disponível em: <https://doi.org/10.1145/3412841.3442037>. Citado na página 101.

BELLOVIN, S. M. Attack surfaces. IEEE Security & Privacy, IEEE, v. 14, n. 3, p. 88–88, 2016. Citado na página 16.

BHARDWAJ, A. et al. Penetration testing framework for smart contract blockchain. Peer-to-Peer Networking and Applications, v. 14, p. 1–16, 09 2021. Citado na página 100.

BHUTA, J.; BOEHM, B.; MEYERS, S. Process elements: Components of software process architectures. In: . [S.l.: s.n.], 2005. v. 3840, p. 332–346. ISBN 978-3-540-31112-6. Citado na página 24.

BILLAWA, P. et al. Sok: Security of microservice applications: A practitioners' perspective on challenges and best practices. In: Proceedings of the 17th International Conference on Availability, Reliability and Security. New York, NY, USA: Association for Computing Machinery, 2022. (ARES '22). ISBN 9781450396707. Disponível em: <https://doi.org/10.1145/3538969.3538986>. Citado na página 101.

BINAMUNGU, L. P.; EMBURY, S. M.; KONSTANTINOU, N. Maintaining behaviour driven development specifications: Challenges and opportunities. In: 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER). [S.l.: s.n.], 2018. p. 175–184. Citado na página 26.

BRADBURY, M. et al. Identifying attack surfaces in the evolving space industry using reference architectures. In: IEEE. 2020 IEEE Aerospace Conference. [S.l.], 2020. p. 1–20. Citado 2 vezes nas páginas 29 e 59.

BRANDOLINI, A. Introducing event storming. blog, Ziobrando's Lair, v. 18, 2013. Citado 5 vezes nas páginas 17, 24, 28, 40 e 41.

BROCKE, J. v.; HEVNER, A.; MAEDCHE, A. Introduction to design science research. In: ____. [S.l.: s.n.], 2020. p. 1–13. ISBN 978-3-030-46780-7. Citado na página 32.

BUDIGIRI, G. Secure and scalable policy management in cloud native networking. In: Proceedings of the 24th International Middleware Conference: Demos, Posters and Doctoral Symposium. New York, NY, USA: Association for Computing

Machinery, 2023. (Middleware '23), p. 11–14. ISBN 9798400704291. Disponível em: <https://doi.org/10.1145/3626564.3629092>. Citado na página 100.

CALDERóN-GóMEZ, H. et al. Telemonitoring system for infectious disease prediction in elderly people based on a novel microservice architecture. IEEE Access, v. 8, p. 118340 – 118354, 06 2020. Citado na página 102.

CASALE, G. et al. Radon: Rational decomposition and orchestration for serverless computing. SICS Software-Intensive Cyber-Physical Systems, v. 35, n. 1, p. 77–87, 2020. ISSN 2524-8529. Disponível em: <https://doi.org/10.1007/s00450-019-00413-w>. Citado na página 100.

CASTRO, J.; LARANJEIRO, N.; VIEIRA, M. Detecting dos attacks in microservice applications: Approachnbsp;and case study. In: Proceedings of the 11th Latin-American Symposium on Dependable Computing. New York, NY, USA: Association for Computing Machinery, 2023. (LADC '22), p. 73–78. ISBN 9781450397377. Disponível em: <https://doi.org/10.1145/3569902.3569916>. Citado na página 96.

CASTRO, J.; LARANJEIRO, N.; VIEIRA, M. Generating realistic attack data for microservices: Framework and case study. In: Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing. New York, NY, USA: Association for Computing Machinery, 2023. (LADC '23), p. 60–69. ISBN 9798400708442. Disponível em: <https://doi.org/10.1145/3615366.3615377>. Citado na página 97.

CHEN, A. et al. Dispersing asymmetric ddos attacks with splitstack. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks. New York, NY, USA: Association for Computing Machinery, 2016. (HotNets '16), p. 197–203. ISBN 9781450346610. Disponível em: <https://doi.org/10.1145/3005745.3005773>. Citado na página 97.

CHEN, C.-A. With Great Abstraction Comes Great Responsibility: Sealing the Microservices Attack Surface . In: 2019 IEEE Secure Development (SecDev). Los Alamitos, CA, USA: IEEE Computer Society, 2019. p. 144–144. Disponível em: <https://doi.ieeecomputersociety.org/10.1109/SecDev.2019.00027>. Citado 2 vezes nas páginas 23 e 103.

CHEN, H. et al. A business-oriented methodology to evaluate the security of software architecture quantitatively. International Journal of Software Engineering and Knowledge Engineering, v. 34, 08 2023. Citado na página 94.

CHEN, L. et al. Research on the security of internet of things based on microservices techniques. In: Proceedings of the 2023 3rd International Conference on Big Data, Artificial Intelligence and Risk Management. New York, NY, USA: Association for Computing Machinery, 2024. (ICBAR '23), p. 509–516. ISBN 9798400716478. Disponível em: <https://doi.org/10.1145/3656766.3656853>. Citado na página 100.

CHIFOR, B.-C.; ARSENI, -C.; BICA, I. Iot cloud security design patterns. In: ____. Big Data Platforms and Applications: Case Studies, Methods, Techniques, and Performance Evaluation. Cham: Springer International Publishing, 2021. p. 113–164. ISBN 978-3-030-38836-2. Disponível em: <https://doi.org/10.1007/978-3-030-38836-2_6>. Citado na página 98.

CHONDAMRONGKUL, N.; SUN, J.; WARREN, I. Automated security analysis for microservice architecture. In: 2020 IEEE International Conference on Software Architecture Companion (ICSA-C). [S.l.: s.n.], 2020. p. 79–82. Citado na página 95.

CHONDAMRONGKUL, N.; SUN, J.; WARREN, I. Formal security analysis for software architecture design: An expressive framework to emerging architectural styles. Science of Computer Programming, v. 206, p. 102631, 2021. ISSN 0167-6423. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167642321000241>. Citado na página 97.

CHRISTOFOROU, A. et al. Supporting the decision of migrating to microservices through multi-layer fuzzy cognitive maps. In: SPRINGER. Service-Oriented Computing: 15th International Conference, ICSOC 2017, Malaga, Spain, November 13–16, 2017, Proceedings. [S.l.], 2017. p. 471–480. Citado na página 16.

DEEPMIND, G. Gemini: Advancing language understanding and generation with deep neural networks. Journal of Artificial Intelligence Research, 2024. Disponível em: <https://deepmind.com/research/publications/gemini-2024>. Citado na página 27.

DEMARCO, T. Structure analysis and system specification. In: ____. Pioneers and Their Contributions to Software Engineering: sd&m Conference on Software Pioneers, Bonn, June 28/29, 2001, Original Historic Contributions. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 255–288. ISBN 978-3-642-48354-7. Disponível em: <https://doi.org/10.1007/978-3-642-48354-7_9>. Citado na página 23.

DRAGONI, N. et al. Microservices: Yesterday, today, and tomorrow. In: ____. Present and Ulterior Software Engineering. Cham: Springer International Publishing, 2017. p. 195–216. ISBN 978-3-319-67425-4. Disponível em: <https://doi.org/10.1007/978-3-319-67425-4_12>. Citado na página 99.

DUARTE, A.; ANTUNES, N. An empirical study of docker vulnerabilities and of static code analysis applicability. In: 2018 Eighth Latin-American Symposium on Dependable Computing (LADC). [S.l.: s.n.], 2018. p. 27–36. Citado na página 95.

DZOGOVIC, B. et al. Zero-trust cybersecurity approach for dynamic 5g network slicing with network service mesh and segment-routing over ipv6. In: 2022 International Conference on Development and Application Systems (DAS). [S.l.: s.n.], 2022. p. 105–114. Citado na página 103.

EDDIN, S. K. et al. Quantum microservices: Transforming software architecture with quantum computing. In: BAROLLI, L. (Ed.). Advanced Information Networking and Applications. Cham: Springer Nature Switzerland, 2024. p. 227–237. ISBN 978-3-031-57942-4. Citado na página 100.

ELSAYED, M.; ZULKERNINE, M. Offering security diagnosis as a service for cloud saas applications. Journal of Information Security and Applications, v. 44, p. 32–48, 2019. ISSN 2214-2126. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214212618302771>. Citado na página 99.

ESPINOZA, A. M. et al. Back to the future: N-versioning of microservices. In: 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). [S.l.: s.n.], 2022. p. 415–427. Citado na página 96.

FUCCI, D. et al. Evaluating software security maturity using owasp samm: Different approaches and stakeholders perceptions. Journal of Systems and Software, v. 214, p. 112062, 2024. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121224001079>. Citado na página 17.

GENFER, P.; ZDUN, U. Avoiding excessive data exposure through microservice apis. In: GEROSTATHOPOULOS, I. et al. (Ed.). Software Architecture. Cham: Springer International Publishing, 2022. p. 3–18. ISBN 978-3-031-16697-6. Citado na página 96.

GEORG, G. et al. Verification and trade-off analysis of security properties in uml system models. IEEE Transactions on Software Engineering, IEEE, v. 36, n. 3, p. 338–356, 2010. Citado na página 29.

GERKING, C.; SCHUBERT, D. Component-Based Refinement and Verification of Information-Flow Security Policies for Cyber-Physical Microservice Architectures . In: 2019 IEEE International Conference on Software Architecture (ICSA). Los Alamitos, CA, USA: IEEE Computer Society, 2019. p. 61–70. Disponível em: <https://doi.ieeecomputersociety.org/10.1109/ICSA.2019.00015>. Citado na página 96.

GHORBANI, M. et al. Distappgaurd: Distributed application behaviour profiling in cloud-based environment. In: Proceedings of the 37th Annual Computer Security Applications Conference. New York, NY, USA: Association for Computing Machinery, 2021. (ACSAC '21), p. 837–848. ISBN 9781450385794. Disponível em: <https://doi.org/10.1145/3485832.3485907>. Citado na página 97.

GIURA, P.; JIM, T. Sapphire: using network gateways for iot security. In: Proceedings of the 8th International Conference on the Internet of Things. New York, NY, USA: Association for Computing Machinery, 2018. (IOT '18). ISBN 9781450365642. Disponível em: <https://doi.org/10.1145/3277593.3277611>. Citado na página 100.

GOPAN, G. et al. The intercorrelation between zero trust, dark web, and its implications in the field of digital forensics. In: . [S.l.: s.n.], 2023. p. 197–203. Citado na página 102.

GOYEL, H.; SWARUP, K. Data integrity attack detection using ensemble-based learning for cyber–physical power systems. IEEE Transactions on Smart Grid, PP, p. 1–1, 01 2022. Citado na página 96.

GUERRA-GARCIA, C. et al. Iso/iec 25012-based methodology for managing data quality requirements in the development of information systems: Towards data quality by design. Data Knowledge Engineering, v. 145, p. 102152, 2023. ISSN 0169-023X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0169023X23000125>. Citado na página 26.

HAINDL, P.; KOCHBERGER, P.; SVEGGEN, M. A systematic literature review of inter-service security threats and mitigation strategies in microservice architectures. IEEE Access, v. 12, p. 90252–90286, 2024. Citado na página 23.

HAOUES, M. et al. A guideline for software architecture selection based on ISO 25010 quality related characteristics. International Journal of System Assurance Engineering and Management, Springer, v. 8, p. 886–909, 2017. Citado na página 68.

HAOYU, W.; HAILI, Z. Basic design principles in software engineering. In: 2012 Fourth International Conference on Computational and Information Sciences. [S.l.: s.n.], 2012. p. 1251–1254. Citado na página 19.

HARRIS, S. et al. Evolution of network enumeration strategies in emulated computer networks. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. New York, NY, USA: Association for Computing Machinery, 2018. (GECCO '18), p. 1640–1647. ISBN 9781450357647. Disponível em: <https://doi.org/10.1145/3205651.3208270>. Citado na página 97.

HASSAN, F.; DAS, S. R.; HUSSAIN, M. Importance of Secure Software Development for the Software Development at Different SDLC Phases. 2023. Citado na página 18.

HATTORI, J.; KATO, S. Sentinel: a fast and memory-efficient serverless architecture for lightweight applications. In: Proceedings of the Eighth International Workshop on Serverless Computing. New York, NY, USA: Association for Computing Machinery, 2022. (WoSC '22), p. 13–18. ISBN 9781450399272. Disponível em: <https://doi.org/10.1145/3565382.3565880>. Citado na página 101.

HERNANDEZ, A. A. et al. Jlearn: An instructional environment for java program composition integrating test-driven development and life-cycle management for software quality assurance. In: 2010 International Conference on Networking and Information Technology. [S.l.: s.n.], 2010. p. 166–170. Citado 2 vezes nas páginas 28 e 31.

HERRMANN, P. Information flow analysis of component-structured applications. In: IEEE. Seventeenth Annual Computer Security Applications Conference. [S.l.], 2001. p. 45–54. Citado 2 vezes nas páginas 29 e 59.

HEVNER, A.; CHATTERJEE, S. Design science research in information systems. In: ____. Design Research in Information Systems: Theory and Practice. Boston, MA: Springer US, 2010. p. 9–22. ISBN 978-1-4419-5653-8. Disponível em: <https://doi.org/10.1007/978-1-4419-5653-8_2>. Citado 2 vezes nas páginas 32 e 33.

IBRAHIM, A.; BOZHINOSKI, S.; PRETSCHNER, A. Attack graph generation for microservice architecture. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. New York, NY, USA: Association for Computing Machinery, 2019. (SAC '19), p. 1235–1242. ISBN 9781450359337. Disponível em: <https://doi.org/10.1145/3297280.3297401>. Citado na página 95.

IDRIS, M.; SYARIF, I.; WINARNO, I. Development of vulnerable web application based on owasp api security risks. In: 2021 International Electronics Symposium (IES). [S.l.: s.n.], 2021. p. 190–194. Citado na página 96.

III, G. F.; EL-SHEIKH, E. Open platform infrastructure for industrial control systems security. In: DAIMI, K.; SADOON, A. A. (Ed.). Proceedings of the Second International Conference on Advances in Computing Research (ACR'24). Cham: Springer Nature Switzerland, 2024. p. 233–243. ISBN 978-3-031-56950-0. Citado na página 99.

INDRASIRI, K.; SIRIWARDENA, P. Microservices security fundamentals: Designing, developing, and deploying. In: ____. [S.l.: s.n.], 2018. p. 313–345. ISBN 978-1-4842-3857-8. Citado na página 99.

JARZĘBOWICZ, A.; WEICHBROTH, P. A qualitative study on non-functional requirements in agile software development. IEEE Access, IEEE, V. 9, p. 40458–40475, 2021. Citado na página 69.

JAYALATH, R. K. et al. Microservices vulnerability analysis: A literature review with empirical insights. IEEE Access, v. 12, p. 155168–155204, 2024. Citado na página 23.

JIA, Y. Design and implementation of distributed archives management system based on microservice architecture. In: 2025 4th International Symposium on Computer Applications and Information Technology (ISCAIT). [S.l.: s.n.], 2025. p. 2168–2171. Citado na página 22.

JIANG, H. et al. Oztrust: An o-ran zero-trust security system. In: 2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). [S.l.: s.n.], 2023. p. 129–134. Citado na página 100.

JÜRJENS, J.; SCHRECK, J.; BARTMANN, P. Model-based security analysis for mobile communications. In: Proceedings of the 30th international conference on Software engineering. [S.l.: s.n.], 2008. p. 683–692. Citado 2 vezes nas páginas 29 e 59.

KASAULI, R. et al. Requirements engineering challenges and practices in large-scale agile system development. Journal of Systems and Software, v. 172, p. 110851, 2021. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121220302417>. Citado na página 69.

KE, H.; WU, H.; YANG, D. Towards evolving security requirements of industrial internet: A layered security architecture solution based on data transfer techniques. In: Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies. New York, NY, USA: Association for Computing Machinery, 2021. (CIAT 2020), p. 504–511. ISBN 9781450387828. Disponível em: <https://doi.org/10.1145/3444370.3444620>. Citado na página 102.

KHAN, Z.; ABBASI, A.; PERVEZ, Z. Blockchain and edge computing–based architecture for participatory smart city applications. Concurrency and Computation: Practice and Experience, v. 32, 11 2019. Citado na página 96.

KIM, S.; KIM, B. J.; LEE, D. H. Prof-gen: Practical study on system call whitelist generation for container attack surface reduction. In: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD). [S.l.: s.n.], 2021. p. 278–287. Citado na página 100.

KIM, Y.; KOO, J.; KIM, U.-M. Vulnerabilities and secure coding for serverless applications on cloud computing. In: Human-Computer Interaction. User Experience and Behavior: Thematic Area, HCI 2022, Held as Part of the 24th HCI International Conference, HCII 2022, Virtual Event, June 26 – July 1, 2022, Proceedings, Part III. Berlin, Heidelberg: Springer-Verlag, 2022. p. 145–163. ISBN 978-3-031-05411-2. Disponível em: <https://doi.org/10.1007/978-3-031-05412-9_10>. Citado na página 103.

KIM, Y.; PARK, C.; SHIN, Y.-y. Security consideration of each layers in a cloud-native environment. In: YOU, I.; KIM, H.; ANGIN, P. (Ed.). Mobile Internet Security. Singapore: Springer Nature Singapore, 2023. p. 231–242. ISBN 978-981-99-4430-9. Citado na página 101.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. v. 2, 01 2007. Citado na página 34.

KITCHENHAM, B. A.; DYBA, T.; JORGENSEN, M. Evidence-based software engineering. In: IEEE. Proceedings. 26th International Conference on Software Engineering. [S.l.], 2004. p. 273–281. Citado 2 vezes nas páginas 45 e 46.

KLIMA, M. et al. Selected code-quality characteristics and metrics for internet of things systems. IEEE Access, v. 10, p. 46144–46161, 2022. Citado na página 101.

LAURETIS, L. D. From monolithic architecture to microservices architecture. In: 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). [S.l.: s.n.], 2019. p. 93–96. Citado na página 21.

LEE, W.; CHOE, Y.; GHOSH, R. Recurrent neural network and convolutional neural network for detection of denial of service attack in microservices. In: . [S.l.: s.n.], 2023. p. 1451–1456. Citado na página 100.

LEVCOVITZ, A.; TERRA, R.; VALENTE, M. T. Towards a technique for extracting microservices from monolithic enterprise systems. ArXiv, abs/1605.03175, 2016. Disponível em: <https://api.semanticscholar.org/CorpusID:13549058>. Citado na página 29.

LI, J. et al. A fast and scalable authentication scheme in iot for smart living. Future Generation Computer Systems, v. 117, p. 125–137, 2021. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X20330302>. Citado na página 95.

LI, X.; CHEN, Y.; LIN, Z. Towards automated inter-service authorization for microservice applications. In: Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos. New York, NY, USA: Association for Computing Machinery, 2019. (SIGCOMM Posters and Demos '19), p. 3–5. ISBN 9781450368865. Disponível em: <https://doi.org/10.1145/3342280.3342288>. Citado na página 102.

LI, X. et al. Automatic policy generation for Inter-Service access control of microservices. In: 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, 2021. p. 3971–3988. ISBN 978-1-939133-24-3. Disponível em: <https://www.usenix.org/conference/usenixsecurity21/presentation/li-xing>. Citado na página 96.

LI, Y. et al. An active security defense strategy for microservices based on deep reinforcement learning. In: 2023 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom). [S.l.: s.n.], 2023. p. 410–415. Citado na página 95.

LI, Z. et al. The serverless computing survey: A technical primer for design architecture. ACM Comput. Surv., Association for Computing Machinery, New York, NY, USA, v. 54, n. 10s, set. 2022. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3508360>. Citado na página 102.

LIM, M. Directly and indirectly synchronous communication mechanisms for client-server systems using event-based asynchronous communication framework. IEEE Access, v. 7, p. 81969–81982, 2019. Citado na página 20.

LIN, C. et al. Autonomic security management for iot smart spaces. ACM Trans. Internet Things, Association for Computing Machinery, New York, NY, USA, v. 2, n. 4, ago. 2021. Disponível em: <https://doi.org/10.1145/3466696>. Citado na página 96.

LIN, S. et al. Invicloak: An end-to-end approach to privacy and performance in web content distribution. In: . [S.l.: s.n.], 2022. p. 1947–1961. Citado na página 98.

LIU, X. et al. Research on attack mechanism using attack surface. In: IEEE. 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA). [S.l.], 2020. p. 137–141. Citado na página 58.

LIU, X. et al. Operating systems for resource-adaptive intelligent software: Challenges and opportunities. ACM Trans. Internet Technol., Association for Computing Machinery,

New York, NY, USA, v. 21, n. 2, mar. 2021. ISSN 1533-5399. Disponível em: <https://doi.org/10.1145/3425866>. Citado na página 100.

LYSNE, O. et al. Vendor malware: Detection limits and mitigation. Computer, v. 49, n. 8, p. 62–69, 2016. Citado na página 102.

MAO, R. et al. Preliminary findings about devsecops from grey literature. In: 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS). [S.l.: s.n.], 2020. p. 450–457. Citado na página 100.

Martínez Saucedo, A. et al. Migration of monolithic systems to microservices: A systematic mapping study. Information and Software Technology, v. 177, p. 107590, 2025. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584924001952>. Citado na página 19.

MAST, K.; ARPACI-DUSSEAU, A. C.; ARPACI-DUSSEAU, R. H. Lambdaobjects: Re-aggregating storage and execution for cloud computing. In: Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems. New York, NY, USA: Association for Computing Machinery, 2022. (HotStorage '22), p. 15–22. ISBN 9781450393997. Disponível em: <https://doi.org/10.1145/3538643.3539751>. Citado na página 98.

MATIAS, M. et al. Evaluating effectiveness and security in microservices architecture. Procedia Computer Science, v. 237, p. 626–636, 2024. ISSN 1877-0509. International Conference on Industry Sciences and Computer Science Innovation. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050924011645>. Citado 2 vezes nas páginas 56 e 97.

MAZEIKA, D.; BUTLERIS, R. Identifying security issues with mbse while rebuilding legacy software systems. In: 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE). [S.l.: s.n.], 2020. p. 83–86. Citado na página 97.

MCALEESE, S. et al. Serverless software engineering – and how to get there. In: YILMAZ, M. et al. (Ed.). Systems, Software and Services Process Improvement. Cham: Springer International Publishing, 2022. p. 75–90. ISBN 978-3-031-15559-8. Citado na página 101.

MEADOWS, C. et al. Sidecar-based path-aware security for microservices. In: Proceedings of the 28th ACM Symposium on Access Control Models and Technologies. New York, NY, USA: Association for Computing Machinery, 2023. (SACMAT '23), p. 157–162. ISBN 9798400701733. Disponível em: <https://doi.org/10.1145/3589608.3594742>. Citado na página 101.

MELEGATI, J.; WANG, X. Case survey studies in software engineering research. In: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). [S.l.: s.n.], 2020. p. 1–12. Citado 3 vezes nas páginas 45, 46 e 47.

MINNA, F. et al. Understanding the security implications of kubernetes networking. IEEE Security Privacy, PP, p. 2–12, 07 2021. Citado na página 102.

MINNA, F.; MASSACCI, F. Sok: Run-time security for cloud microservices. are we there yet? Computers Security, v. 127, p. 103119, 04 2023. Citado na página 101.

MOE, M. Comparative study of test-driven development tdd, behavior-driven development bdd and acceptance test–driven development atdd. International Journal of Trend in Scientific Research and Development, Volume-3, p. 231–234, 06 2019. Citado na página 26.

MUDIMBA, M.; PHIRI, J. Decoupling of heterogeneous systems using the microservices architectural model in higher institutions of learning in zambia. In: SILHAVY, R.; SILHAVY, P. (Ed.). Software Engineering Methods in Systems and Network Systems. Cham: Springer Nature Switzerland, 2024. p. 464–481. ISBN 978-3-031-54813-0. Citado na página 20.

NAM, J. et al. Secure inter-container communications using xdp/ebpf. IEEE/ACM Transactions on Networking, v. 31, n. 2, p. 934–947, 2023. Citado na página 101.

NAMBIAR, S. et al. Multilevel secure container deployment framework in edge computing. In: THAMPI, S. M. et al. (Ed.). Security in Computing and Communications. Singapore: Springer Singapore, 2021. p. 49–61. ISBN 978-981-16-0422-5. Citado na página 99.

NEMMINI, S. et al. Fortifying information security: Security implications of microservice and monolithic architectures. Proceedings of the 2023 International Conference on Security of Information and Networks (SIN), 2023. Citado na página 97.

NKOMO, P.; COETZEE, M. Development activities, tools and techniques of secure microservices compositions. In: HENG, S.-H.; LOPEZ, J. (Ed.). Information Security Practice and Experience. Cham: Springer International Publishing, 2019. p. 423–433. ISBN 978-3-030-34339-2. Citado na página 96.

NKOMO, P.; COETZEE, M. Software development activities for secure microservices. In: MISRA, S. et al. (Ed.). Computational Science and Its Applications – ICCSA 2019. Cham: Springer International Publishing, 2019. p. 573–585. ISBN 978-3-030-24308-1. Citado na página 101.

NORTH, D. Introducing BDD. 2006. Https://dannorth.net/introducing-bdd/. Citado 3 vezes nas páginas 26, 69 e 70.

OLSSON, T.; SENTILLES, S.; PAPATHEOCHAROUS, E. A Systematic Literature Review of empirical research on quality requirements. Requirements Engineering. Springer, Springer, V.27, n. 2, p. 249–271, 2022. Citado 3 vezes nas páginas 30, 69 e 70.

OMMERING, R. van. Independent deployment - the reason behind components? In: . [S.l.: s.n.], 2001. Citado na página 21.

OpenAI. ChatGPT: Overview. 2024. Acesso em: 01 ago. 2024. Disponível em: <https://www.openai.com/blog/chatgpt>. Citado na página 27.

OPPERMANN, A. et al. Secure cloud computing : Risk analysis for secure cloud reference architecture in legal metrology. In: 2018 Federated Conference on Computer Science and Information Systems (FedCSIS). [S.l.: s.n.], 2018. p. 593–602. Citado na página 100.

OSMAN, A.; BORN, J.; STRUFE, T. Mitigating internal, stealthy dos attacks in microservice networks. In: JOHNEN, C.; SCHILLER, E. M.; SCHMID, S. (Ed.). Stabilization, Safety, and Security of Distributed Systems. Cham: Springer International Publishing, 2021. p. 500–504. ISBN 978-3-030-91081-5. Citado na página 99.

OTTERSTAD, C.; YARYGINA, T. Low-level exploitation mitigation by diverse microservices. In: . [S.l.: s.n.], 2017. p. 49–56. ISBN 978-3-319-67261-8. Citado na página 98.

OUMOUSSA, I.; FAIEQ, S.; SAIDI, R. Microservices: Investigating underpinnings. In: AHMED, M. B. et al. (Ed.). Emerging Trends in Intelligent Systems & Network Security. Cham: Springer International Publishing, 2023. p. 343–351. ISBN 978-3-031-15191-0. Citado na página 99.

PAI, B.; HEBBAR, A.; KUMAR, M. Securing microservice-driven applications based on api access graphs using supervised machine learning techniques. In: ____. [S.l.: s.n.], 2023. p. 587–598. ISBN 978-981-19-6633-0. Citado 2 vezes nas páginas 56 e 101.

PEREIRA-VALE, A. et al. Security in microservice-based systems: A multivocal literature review. Computers Security, v. 103, p. 25, 01 2021. Citado na página 101.

PETERSEN, K. Guidelines for case survey research in software engineering. Contemporary empirical methods in software engineering, Springer, p. 63–92, 2020. Citado 2 vezes nas páginas 46 e 47.

PONCE, F. et al. Microservices security: Bad vs. good practices. In: Software Architecture. ECSA 2022 Tracks and Workshops: Prague, Czech Republic, September 19–23, 2022, Revised Selected Papers. Berlin, Heidelberg: Springer-Verlag, 2022. p. 337–352. ISBN 978-3-031-36888-2. Disponível em: <https://doi.org/10.1007/978-3-031-36889-9_23>. Citado na página 99.

PONCE, F. et al. To security and beyond: On the impacts of microservice security smells and refactorings. In: 2023 XLIX Latin American Computer Conference (CLEI). [S.l.: s.n.], 2023. p. 1–10. Citado na página 102.

QIANG, W. et al. Intrinsic security and self-adaptive cooperative protection enabling cloud native network slicing. IEEE Transactions on Network and Service Management, v. 18, n. 2, p. 1287–1304, 2021. Citado na página 98.

RADEMACHER, F.; SORGALLA, J.; SACHWEH, S. Challenges of domain-driven microservice design: A model-driven perspective. IEEE Software, v. 35, n. 3, p. 36–43, 2018. Citado 2 vezes nas páginas 14 e 42.

RADFORD, A. et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2019. Acesso em: 10 ago. 2024. Disponível em: <https://arxiv.org/abs/2005.14165>. Citado na página 27.

RAHARJANA, I. K.; HARRIS, F.; JUSTITIA, A. Tool for generating behavior-driven development test-cases. Journal of Information Systems Engineering and Business Intelligence, v. 6, n. 1, p. 27–36, Apr. 2020. Disponível em: <https://e-journal.unair.ac.id/JISEBI/article/view/18082>. Citado na página 26.

RAJ, V.; SADAM, R. Performance and complexity comparison of service oriented architecture and microservices architecture. International Journal of Communication Networks and Distributed Systems, v. 27, n. 1, p. 100–117, 2021. Disponível em: <https://www.inderscienceonline.com/doi/abs/10.1504/IJCNDS.2021.116463>. Citado na página 21.

RAJBHOJ, A. et al. Accelerating software development using generative ai: Chatgpt case study. In: Proceedings of the 17th Innovations in Software Engineering Conference. [S.l.: s.n.], 2024. p. 1–11. Citado 5 vezes nas páginas 7, 29, 51, 54 e 69.

RAMIREZ, A.; AIELLO, A.; LINCKE, S. J. A survey and comparison of secure software development standards. In: 2020 13th CMI Conference on Cybersecurity and Privacy (CMI) - Digital Transformation - Potentials and Challenges(51275). [S.l.: s.n.], 2020. p. 1–6. Citado na página 13.

RAMOS-CRUZ, B.; ANDREU-PEREZ, J.; MARTíNEZ, L. The cybersecurity mesh: A comprehensive survey of involved artificial intelligence methods, cryptographic protocols and challenges for future research. Neurocomputing, v. 581, p. 127427, 2024. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S092523122400198X>. Citado na página 102.

RASTOGI, V. et al. Cimplifier: automatically debloating containers. In: . [S.l.: s.n.], 2017. p. 476–486. ISBN 978-1-4503-5105-8. Citado na página 96.

RESELMAN, B. 5 design principles for microservices | Red Hat Developer. 2022. <https://developers.redhat.com/articles/2022/01/11/5-design-principles-microservices#>. (Accessed on 05/05/2023). Citado 3 vezes nas páginas 28, 31 e 41.

RESELMAN, B. The disadvantages vs. benefits of microservices | Red Hat Developer. 2022. <https://developers.redhat.com/articles/2022/01/25/disadvantages-microservices>. (Accessed on 05/05/2023). Citado 2 vezes nas páginas 28 e 31.

RESELMAN, B. Implementing and using microservices | Red Hat Developer. 2022. <https://developers.redhat.com/articles/2022/01/19/monolith-microservices-how-applications-evolve#creating_a_microservices_oriented_application>. (Accessed on 05/05/2023). Citado 2 vezes nas páginas 28 e 31.

Rezaei Nasab, A. et al. An empirical study of security practices for microservices systems. Journal of Systems and Software, v. 198, p. 111563, 2023. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121222002394>. Citado 3 vezes nas páginas 17, 28 e 31.

REZAPOUR, R. et al. Security in fog computing: A systematic review on issues, challenges and solutions. Computer Science Review, Elsevier, v. 41, p. 100421, 2021. Citado na página 29.

ROCHA, F.; SOARES, M.; RODRIGUEZ, G. Patterns in microservices-based development: A grey literature review. In: Anais do XXVI Congresso Ibero-Americano em Engenharia de Software. Porto Alegre, RS, Brasil: SBC, 2023. p. 61–76. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/cibse/article/view/24693>. Citado na página 19.

ROCHA, F. G.; MISRA, S.; SOARES, M. S. Guidelines for future agile methodologies and architecture reconciliation for software-intensive systems. Electronics, v. 12, n. 7, 2023. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/12/7/1582>. Citado na página 24.

RODRIGUEZ, G. et al. Understanding and addressing the allocation of microservices into containers: A review. IETE Journal of Research, Taylor Francis, v. 0, n. 0, p. 1–14, 2023. Disponível em: <https://doi.org/10.1080/03772063.2023.2205864>. Citado na página 19.

ROMAN, R.; LOPEZ, J.; MAMBO, M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Future Generation Computer Systems, Elsevier, v. 78, p. 680–698, 2018. Citado na página 29.

RON, J. et al. Highly available blockchain nodes with n-version design. IEEE Trans. Dependable Secur. Comput., IEEE Computer Society Press, Washington, DC, USA, v. 21, n. 4, p. 4084–4097, jul. 2024. ISSN 1545-5971. Disponível em: <https://doi.org/10.1109/TDSC.2023.3346195>. Citado na página 97.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering. Springer, Springer, V.14, p. 131–164, 2009. Citado 2 vezes nas páginas 46 e 70.

SADIQ, A. et al. Detection of denial of service attack in cloud based kubernetes using ebpf. Applied Sciences, v. 13, n. 8, 2023. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/13/8/4700>. Citado na página 96.

SAEED, H. et al. Review of techniques for integrating security in software development lifecycle. Computers, Materials and Continua, v. 82, n. 1, p. 139–172, 2025. ISSN 1546-2218. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1546221825000128>. Citado na página 18.

SäNGER, N.; ABECK, S. Fine-grained authorization in microservice architecture: A decentralized approach. In: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing. New York, NY, USA: Association for Computing Machinery, 2024. (SAC '24), p. 1219–1222. ISBN 9798400702433. Disponível em: <https://doi.org/10.1145/3605098.3636121>. Citado na página 97.

SANKARAN, A.; DATTA, P.; BATES, A. Workflow integration alleviates identity and access management in serverless computing. In: Proceedings of the 36th Annual Computer Security Applications Conference. New York, NY, USA: Association for Computing Machinery, 2020. (ACSAC '20), p. 496–509. ISBN 9781450388580. Disponível em: <https://doi.org/10.1145/3427228.3427665>. Citado na página 103.

SANTOS, S. et al. Using Behavior-Driven Development (BDD) for Non-functional Requirements. 2024. Citado na página 46.

SANTOS, S. et al. Using behavior-driven development (bdd) for non-functional requirements. Software, v. 3, n. 3, p. 271–283, 2024. ISSN 2674-113X. Disponível em: <https://www.mdpi.com/2674-113X/3/3/14>. Citado 2 vezes nas páginas 30 e 69.

SCACCHI, W.; ALSPAUGH, T. A. Securing software ecosystem architectures: Challenges and opportunities. IEEE Software, v. 36, n. 3, p. 33–38, 2019. Citado na página 13.

SCHMIDT, R. A.; THIRY, M. Microservices identification strategies : A review focused on model-driven engineering and domain driven design approaches. In: 2020 15th Iberian Conference on Information Systems and Technologies (CISTI). [S.l.: s.n.], 2020. p. 1–6. Citado na página 14.

SHAH, H.; CHUNG, K. Archie cochrane and his vision for evidence-based medicine. Plastic and Reconstructive Surgery, v. 124, n. 3, p. 982–988, set. 2009. Citado na página 34.

SHARMA, S. et al. Kubernetes continuous development. In: MAHAPATRA, R. P. et al. (Ed.). Proceedings of International Conference on Recent Trends in Computing. Singapore: Springer Nature Singapore, 2022. p. 559–568. ISBN 978-981-16-7118-0. Citado na página 98.

SHEN, Z. et al. X-containers: Breaking down barriers to improve performance and isolation of cloud-native containers. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. New York, NY, USA: Association for Computing Machinery, 2019. (ASPLOS '19), p. 121–135. ISBN 9781450362405. Disponível em: <https://doi.org/10.1145/3297858.3304016>. Citado na página 103.

SHEVCHUK, D. et al. Designing secured services for authentication, authorization, and accounting of users. In: CEUR Workshop Proceedings. [S.l.: s.n.], 2023. v. 3550, p. 217–225. ISSN 1613-0073. Conference Paper. Citado na página 96.

SHEVCHUK, R. et al. Software for improve the security of kubernetes-based ci/cd pipeline. In: 2023 13th International Conference on Advanced Computer Information Technologies (ACIT). [S.l.: s.n.], 2023. p. 420–425. Citado na página 101.

SILVA, M. V. S. et al. Guidelines for the application of event driven architecture in micro services with high volume of data. In: Proceedings of the 27th International Conference on Enterprise Information Systems - Volume 2. [S.l.: s.n.], 2025. p. 859–866. ISBN 978-989-758-749-8. ISSN 2184-4992. Citado na página 13.

SILVA, R.; BRITO, A.; FILHO, J. P. de L. Automation of security controls for continuous compliance in vulnerability management. In: Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing. New York, NY, USA: Association for Computing Machinery, 2024. (LADC '24), p. 1–10. ISBN 9798400717406. Disponível em: <https://doi.org/10.1145/3697090.3697107>. Citado na página 95.

STALLENBERG, D. M.; PANICHELLA, A. Jcomix: a search-based tool to detect xml injection vulnerabilities in web applications. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York, NY, USA: Association for Computing Machinery, 2019. (ESEC/FSE 2019), p. 1090–1094. ISBN 9781450355728. Disponível em: <https://doi.org/10.1145/3338906.3341178>. Citado na página 98.

STEINDL, G. et al. Toward semantic event-handling for building explainable cyber-physical systems. IEEE Open Journal of the Industrial Electronics Society, v. 5, p. 928–945, 2024. Citado na página 26.

STERGIOPOULOS, G.; DEDOUSIS, P.; GRITZALIS, D. Automatic analysis of attack graphs for risk mitigation and prioritization on large-scale and complex networks in industry 4.0. International Journal of Information Security, v. 21, n. 1, p. 37–59, 2022. ISSN 1615-5270. Disponível em: <https://doi.org/10.1007/s10207-020-00533-4>. Citado na página 95.

SUN, W. et al. Heuristic network security risk assessment based on attack graph. In: KHOSRAVI, M. R.; HE, Q.; DAI, H. (Ed.). Cloud Computing. Cham: Springer International Publishing, 2022. p. 181–194. ISBN 978-3-030-99191-3. Citado na página 97.

SUNEJA, S.; KANSO, A.; ISCI, C. Can container fusion be securely achieved? In: Proceedings of the 5th International Workshop on Container Technologies and Container Clouds. New York, NY, USA: Association for Computing Machinery, 2019. (WOC '19), p. 31–36. ISBN 9781450370332. Disponível em: <https://doi.org/10.1145/3366615.3368356>. Citado na página 96.

TALEB, N. N. Antifragile: Things That Gain from Disorder. [S.l.]: Random House, 2012. Citado na página 22.

Technical Committee: ISO/IEC JTC 1/SC 7. ISO/IEC 25010:2023. 2023. Https://www.iso.org/standard/78176.html. Citado na página 25.

THEISEN, C. et al. Risk-based attack surface approximation: how much data is enough? In: IEEE. 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP). [S.l.], 2017. p. 273–282. Citado na página 58.

THEODOROPOULOS, T. et al. Security in cloud-native services: A survey. Journal of Cybersecurity and Privacy, v. 3, n. 4, p. 758–793, 2023. ISSN 2624-800X. Disponível em: <https://www.mdpi.com/2624-800X/3/4/34>. Citado na página 101.

TORKURA, K. et al. Cavas: Neutralizing application and container security vulnerabilities in the cloud native era. In: . [S.l.: s.n.], 2018. Citado na página 96.

TORKURA, K. A. et al. A cyber risk based moving target defense mechanism for microservice architectures. In: 2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). [S.l.: s.n.], 2018. p. 932–939. Citado na página 95.

TORKURA, K. A.; SUKMANA, M. I.; MEINEL, C. Integrating continuous security assessments in microservices and cloud native applications. In: Proceedings of The10th International Conference on Utility and Cloud Computing. New York, NY, USA: Association for Computing Machinery, 2017. (UCC '17), p. 171–180. ISBN 9781450351492. Disponível em: <https://doi.org/10.1145/3147213.3147229>. Citado na página 98.

TORQUATO, M.; VIEIRA, M. Moving target defense in cloud computing: A systematic mapping study. Computers Security, v. 92, p. 101742, 2020. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404820300286>. Citado na página 99.

TOURANI, R. et al. Towards security-as-a-service in multi-access edge. In: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing. New York, NY, USA: Association for Computing Machinery, 2019. (SEC '19), p. 358–363. ISBN 9781450367332. Disponível em: <https://doi.org/10.1145/3318216.3363335>. Citado na página 102.

TOZER, B.; MAZZUCHI, T.; SARKANI, S. Optimizing attack surface and configuration diversity using multi-objective reinforcement learning. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). [S.l.: s.n.], 2015. p. 144–149. Citado na página 100.

TRAN, A.-D.; YSKOUT, K.; JOOSEN, W. Andras: Automated attack surface extraction for android applications. In: 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS). [S.l.: s.n.], 2023. p. 406–417. Citado na página 95.

UMEUGO, W. Secure software development lifecycle: A case for adoption in software smes. International Journal of Advanced Research in Computer Science, v. 14, p. 5–12, 02 2023. Citado na página 18.

UNSEL, V. et al. Risk-based authentication for openstack: A fully functional implementation and guiding example. In: Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy. New York, NY, USA: Association for Computing Machinery, 2023. (CODASPY '23), p. 237–243. ISBN 9798400700675. Disponível em: <https://doi.org/10.1145/3577923.3583634>. Citado na página 100.

VASSILIOU-GIOLES, T. Solving the instance identification problem in micro-service testing. In: CLARK, D.; MENENDEZ, H.; CAVALLI, A. R. (Ed.). Testing Software and Systems. Cham: Springer International Publishing, 2022. p. 189–195. ISBN 978-3-031-04673-5. Citado na página 102.

VERDERAME, L. et al. Secco: Automated services to secure containers in the devops paradigm. In: Proceedings of the 2023 International Conference on Research in Adaptive and Convergent Systems. New York, NY, USA: Association for Computing Machinery, 2023. (RACS '23). ISBN 9798400702280. Disponível em: <https://doi.org/10.1145/3599957.3606222>. Citado na página 100.

WALKER, A.; LAIRD, I.; CERNY, T. On automatic software architecture reconstruction of microservice applications. In: KIM, H.; KIM, K. J.; PARK, S. (Ed.). Information Science and Applications. Singapore: Springer Singapore, 2021. p. 223–234. ISBN 978-981-33-6385-4. Citado na página 99.

WANG, H.; WANG, Y.; HEMALATHA, K. L. Security technology in microservice architecture. In: JANSEN, B. J.; ZHOU, Q.; YE, J. (Ed.). Proceedings of the 3rd International Conference on Cognitive Based Information Processing and Applications—Volume 2. Singapore: Springer Nature Singapore, 2024. p. 69–79. ISBN 978-981-97-1979-2. Citado na página 101.

WANG, Y.; KADIYALA, H.; RUBIN, J. Promises and challenges of microservices: an exploratory study. Empirical Softw. Engg., Kluwer Academic Publishers, USA, v. 26, n. 4, jul. 2021. ISSN 1382-3256. Disponível em: <https://doi.org/10.1007/s10664-020-09910-y>. Citado 2 vezes nas páginas 56 e 100.

WASEEM, M. et al. On the nature of issues in five open source microservices systems: An empirical study. In: Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering. New York, NY, USA: Association for Computing Machinery, 2021. (EASE '21), p. 201–210. ISBN 9781450390538. Disponível em: <https://doi.org/10.1145/3463274.3463337>. Citado na página 99.

WEERASINGHE, L.; PERERA, I. Evaluating the inter-service communication on microservice architecture. In: 2022 7th International Conference on Information Technology Research (ICITR). [S.l.: s.n.], 2022. p. 1–6. Citado na página 20.

WEN, J. et al. Rise of the planet of serverless computing: A systematic review. ACM Trans. Softw. Eng. Methodol., Association for Computing Machinery, New York, NY, USA, v. 32, n. 5, jul. 2023. ISSN 1049-331X. Disponível em: <https://doi.org/10.1145/3579643>. Citado na página 100.

WIZENTY, P. et al. Model-driven security smell resolution in microservice architecture using lemma. In: FILL, H.-G. et al. (Ed.). Software Technologies. Cham: Springer Nature Switzerland, 2024. p. 29–49. ISBN 978-3-031-61753-9. Citado na página 99.

WOHLIN, C.; HÖST, M.; HENNINGSSON, K. Empirical research methods in software engineering. Empirical methods and studies in software engineering: Experiences from ESERNET. Springer, Springer, p. 7–23, 2003. Citado na página 46.

WOHLIN, C. et al. Experimentation in software engineering. [S.l.]: Springer, 2012. v. 236. Citado na página 47.

WOLFF, E. [S.l.: s.n.], 2019. Citado na página 21.

WONG, A. Y. et al. On the security of containers: Threat modeling, attack analysis, and mitigation strategies. Computers Security, v. 128, p. 103140, 2023. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404823000500>. Citado na página 99.

XU, H. et al. Multi-dimensional moving target defense method based on adaptive simulated annealing genetic algorithm. Electronics, v. 13, n. 3, 2024. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/13/3/487>. Citado na página 99.

XU, M. et al. Zero-trust security authentication based on spa and endogenous security architecture. Electronics, v. 12, n. 4, 2023. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/12/4/782>. Citado 2 vezes nas páginas 56 e 103.

XU, S. et al. Log2policy: An approach to generate fine-grained access control rules for microservices from scratch. In: Proceedings of the 39th Annual Computer Security Applications Conference. New York, NY, USA: Association for Computing Machinery, 2023. (ACSAC '23), p. 229–240. ISBN 9798400708862. Disponível em: <https://doi.org/10.1145/3627106.3627137>. Citado na página 98.

YANG, F. et al. Endoprocess: Programmable and extensible subprocess isolation. In: Proceedings of the 2023 New Security Paradigms Workshop. New York, NY, USA: Association for Computing Machinery, 2023. (NSPW '23), p. 92–101. ISBN 9798400716201. Disponível em: <https://doi.org/10.1145/3633500.3633507>. Citado na página 97.

YANG, Y. et al. Security challenges in the container cloud. In: 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA). [S.l.: s.n.], 2021. p. 137–145. Citado na página 101.

YIN, K.; DU, Q. On Representing Resilience Requirements of Microservice Architecture Systems. 2020. Disponível em: <https://arxiv.org/abs/1909.13096>. Citado 3 vezes nas páginas 22, 56 e 99.

YIN, R. K. Case study methods. In: . [S.l.]: American Psychological Association, 2012. Citado na página 46.

ZENG, Q. et al. Full-stack vulnerability analysis of the cloud-native platform. Computers Security, v. 129, p. 103173, 03 2023. Citado na página 97.

ZHANG, W. Security monitoring of microservice-based applications. TechRxiv, 2023. Citado 3 vezes nas páginas 28, 31 e 59.

ZHANG, X. et al. Differentiated security requirements: An exploration of microservice placement algorithms in internet of vehicles. Electronics, v. 13, n. 8, 2024. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/13/8/1597>. Citado na página 97.

ZHOU, X. et al. A map of threats to validity of Systematic Literature Reviews in Software Engineering. In: IEEE. 2016 23rd Asia-Pacific Software Engineering Conference (APSEC). [S.l.], 2016. p. 153–160. Citado na página 70.

ZHU, H.; GEHRMANN, C.; ROTH, P. Access security policy generation for containers as a cloud service. SN Computer Science, v. 4, n. 6, p. 748, 2023. ISSN 2661-8907. Disponível em: <https://doi.org/10.1007/s42979-023-02186-1>. Citado na página 95.

ZIMMERMANN, O. et al. Interface representation patterns: Crafting and consuming message-based remote apis. In: Proceedings of the 22nd European Conference on Pattern Languages of Programs. New York, NY, USA: Association for Computing Machinery, 2017. (EuroPLoP '17). ISBN 9781450348485. Disponível em: <https://doi.org/10.1145/3147704.3147734>. Citado na página 98.

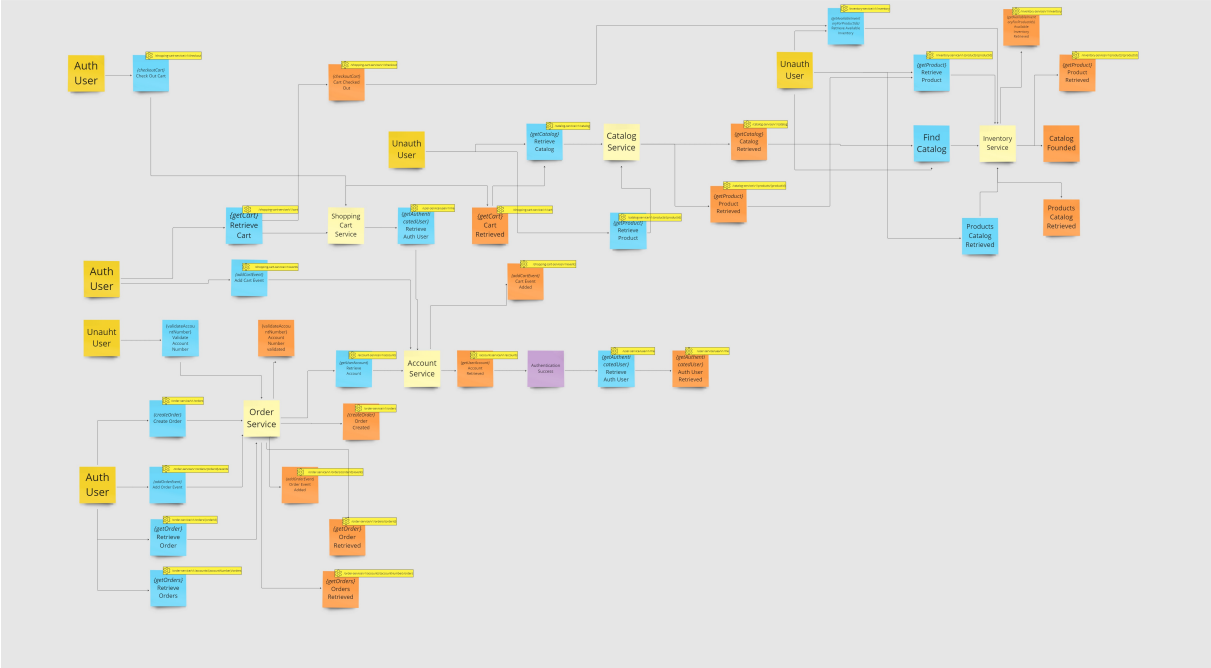


Figura 16 – Visão geral do Cenário atual

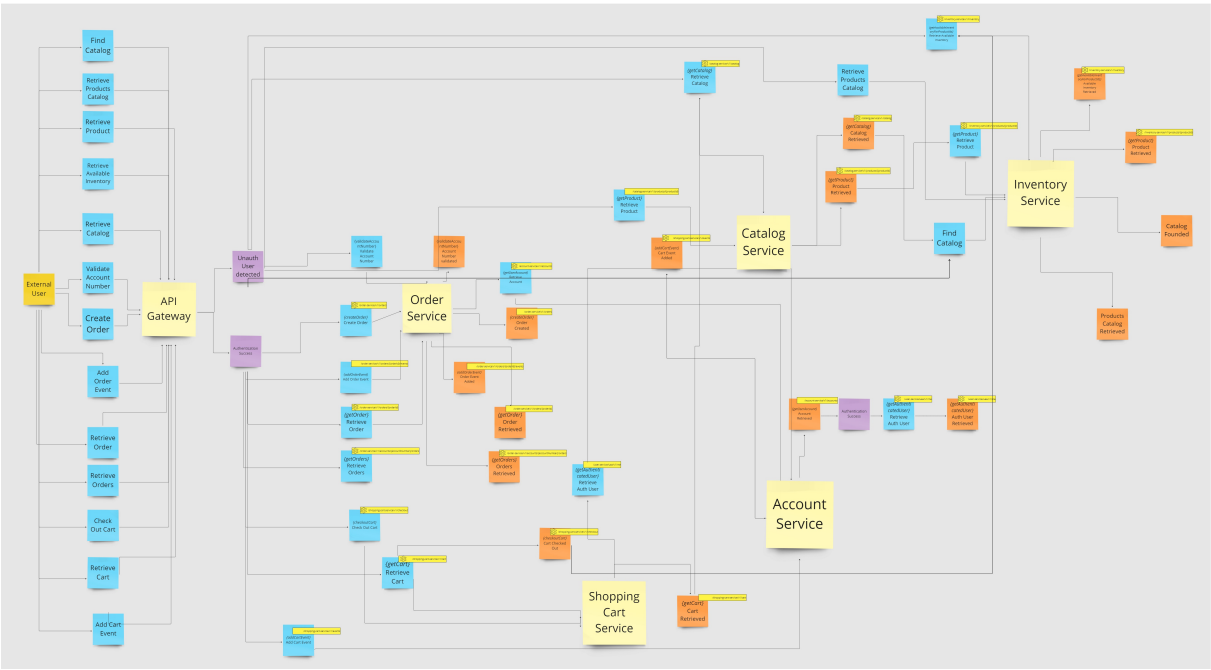


Figura 17 – Visão geral do Cenário Esperado

7.3 Imagens de visão geral

Tabela 13 – Artigos Selecionados do Mapeamento Sistemático da Literatura

Artigo	**Título**
(CHEN et al., 2023)	*A Business-Oriented Methodology to Evaluate the Security of Software Architecture Quantitatively*

Tabela 13 Continuação

Artigo	Título
(BANGUI; ROSSI; BUHNOVA, 2022)	*A Conceptual Antifragile Microservice Framework for Reshaping Critical Infrastructures*
(TORKURA et al., 2018b)	*A Cyber Risk Based Moving Target Defense Mechanism for Microservice Architectures*
(LI et al., 2021)	*A Fast and Scalable Authentication Scheme in IoT for Smart Living*
(ALKADI; MOUSTAFA; TURNBULL, 2020)	*A Review of Intrusion Detection and Blockchain Applications in the Cloud: Approaches, Challenges and Solutions*
(ZHU; GEHRMANN; ROTH, 2023)	*Access Security Policy Generation for Containers as a Cloud Service*
(LI et al., 2023)	*An Active Security Defense Strategy for Microservices based on Deep Reinforcement Learning*
(ALKADI; MOUSTAFA; TURNBULL, 2020)	*An Advanced DevOps Environment for Microservice-based Applications*
(DUARTE; ANTUNES, 2018)	*An Empirical Study of Docker Vulnerabilities and of Static Code Analysis Applicability*
(TRAN; YSKOUT; JOOSEN, 2023)	*AndrAS: Automated Attack Surface Extraction for Android Applications*
(IBRAHIM; BOZHINOSKI; PRETSCHNER, 2019)	*Attack graph generation for microservice architecture*
(SILVA; BRITO; FILHO, 2024)	*Automation of Security Controls for Continuous Compliance in Vulnerability Management*
(CHONDAMRONGKUL; SUN; WARREN, 2020)	*Automated Security Analysis for Microservice Architecture*
(STERGIOPOULOS; DEDOUSIS; GRITZALIS, 2022)	*Automatic analysis of attack graphs for risk mitigation and prioritization on large-scale and complex networks in Industry 4.0*

Tabela 13 Continuação

Artigo	Título
(LI et al., 2021)	*Automatic policy generation for inter-service access control of microservices*
(LIN et al., 2021)	*Autonomic Security Management for IoT Smart Spaces*
(GENFER; ZDUN, 2022)	*Avoiding Excessive Data Exposure Through Microservice APIs*
(ESPINOZA et al., 2022)	*Back to the future: N-Versioning of Microservices*
(KHAN; ABBASI; PERVEZ, 2019) (SUNEJA; KANSO; ISCI, 2019)	*Blockchain and edge computing–based architecture for participatory smart city applications* *Can Container Fusion Be Securely Achieved?*
(TORKURA et al., 2018a)	*CAVAS: Neutralizing Application and Container Security Vulnerabilities in the Cloud Native Era*
(BEEKMAN; PORTER, 2017)	*Challenges For Scaling Applications Across Enclaves*
(RASTOGI et al., 2017)	*Cimplifier: automatically debloating containers*
(GERKING; SCHUBERT, 2019)	*Component-based refinement and verification of information-flow security policies for cyber-physical microservice architectures*
(GOYEL; SWARUP, 2022)	*Data Integrity Attack Detection Using Ensemble-Based Learning for Cyber-Physical Power Systems*
(SHEVCHUK et al., 2023a)	*Designing Secured Services for Authentication, Authorization, and Accounting of Users*
(CASTRO; LARANJEIRO; VIEIRA, 2023a)	*Detecting DoS Attacks in Microservice Applications: Approach and Case Study*
(SADIQ et al., 2023)	*Detection of Denial of Service Attack in Cloud Based Kubernetes Using eBPF*
(NKOMO; COETZEE, 2019a)	*Development Activities, Tools and Techniques of Secure Microservices Compositions*
(IDRIS; SYARIF; WINARNO, 2021)	*Development of Vulnerable Web Application Based on OWASP API Security Risks*

Tabela 13 Continuação

Artigo	Título
(ZHANG et al., 2024)	*Differentiated Security Requirements: An Exploration of Microservice Placement Algorithms in Internet of Vehicles*
(CHEN et al., 2016)	*Dispersing Asymmetric DDoS Attacks with SplitStack*
(GHORBANI et al., 2021)	*DistAppGaurd: Distributed Application Behaviour Profiling in Cloud-Based Environment*
(YANG et al., 2023)	*Endoprocess: Programmable and Extensible Subprocess Isolation*
(ADAMOS et al., 2023)	*Enhancing attack resilience of cyber-physical systems through state dependency graph models*
(ANTUNES, 2024)	*Evaluating intrusion detection for microservice applications: Benchmark, dataset, and case studies*
(MATIAS et al., 2024)	*Evaluating Effectiveness and Security in Microservices Architecture*
(ARAUJO; ANTUNES; VIEIRA, 2023a)	*Evaluation of Machine Learning for Intrusion Detection in Microservice Applications*
(HARRIS et al., 2018)	*Evolution of network enumeration strategies in emulated computer networks*
(SäNGER; ABECK, 2024)	*Fine-Grained Authorization in Microservice Architecture: A Decentralized Approach*
(CHONDAMRONGKUL; SUN; WARREN, 2021)	*Formal security analysis for software architecture design: An expressive framework to emerging architectural styles*
(NEMMINI et al., 2023)	*Fortifying Information Security: Security Implications of Microservice and Monolithic Architectures*
(ZENG et al., 2023)	*Full-stack Vulnerability Analysis of the Cloud-native Platform*
(CASTRO; LARANJEIRO; VIEIRA, 2023b)	*Generating Realistic Attack Data for Microservices: Framework and Case Study*
(SUN et al., 2022)	*Heuristic Network Security Risk Assessment Based on Attack Graph*
(RON et al., 2024)	*Highly Available Blockchain Nodes With N-Version Design*
(MAZEIKA; BUTLERIS, 2020)	*Identifying Security Issues with MBSE while Rebuilding Legacy Software Systems*

Tabela 13 Continuação

Artigo	Título
(BEHRAD et al., 2021)	*Impacts of Service Decomposition Models on Security Attributes: A Case Study with 5G Network Repository Function*
(TORKURA; SUKMANA; MEINEL, 2017)	*Integrating Continuous Security Assessments in Microservices and Cloud Native Applications*
(ZIMMERMANN et al., 2017)	*Interface Representation Patterns: Crafting and Consuming Message-Based Remote APIs*
(QIANG et al., 2021)	*Intrinsic Security and Self-Adaptive Cooperative Protection Enabling Cloud Native Network Slicing*
(ARAUJO; ANTUNES; VIEIRA, 2023b)	*Intrusion Detection and Tolerance for Microservice Applications*
(LIN et al., 2022)	*InviCloak: An End-to-End Approach to Privacy and Performance in Web Content Distribution*
(CHIFOR; ARSENI; BICA, 2021)	*IoT Cloud Security Design Patterns*
(STALLENBERG; PANICHELLA, 2019)	*JCOMIX: a search-based tool to detect XML injection vulnerabilities in web applications*
(SHARMA et al., 2022)	*Kubernetes Continuous Development*
(MAST; ARPACI-DUSSEAU; ARPACI-DUSSEAU, 2022)	*LambdaObjects: Re-aggregating storage and execution for cloud computing*
(BANDARA et al., 2021)	*Let'sTrace — Blockchain, Federated Learning and TUF/In-ToTo Enabled Cyber Supply Chain Provenance Platform*
(XU et al., 2023b)	*Log2Policy: An Approach to Generate Fine-Grained Access Control Rules for Microservices from Scratch*
(OTTERSTAD; YARYGINA, 2017)	*Low-Level Exploitation Mitigation by Diverse Microservices*

Tabela 13 Continuação

Artigo	Título
(INDRASIRI; SIRIWARDENA, 2018)	*Microservices Security Fundamentals: Designing, Developing, and Deploying*
(PONCE et al., 2022)	*Microservices Security: Bad vs. Good Practices*
(OUMOUSSA; FAIEQ; SAIDI, 2023)	*Microservices: Investigating Underpinnings*
(DRAGONI et al., 2017)	*Microservices: Yesterday, Today, and Tomorrow*
(OSMAN; BORN; STRUFE, 2021)	*Mitigating Internal, Stealthy DoS Attacks in Microservice Networks*
(WIZENTY et al., 2024)	*Model-Driven Security Smell Resolution in Microservice Architecture Using LEMMA*
(TORQUATO; VIEIRA, 2020)	*Moving target defense in cloud computing: A systematic mapping study*
(XU et al., 2024)	*Multi-Dimensional Moving Target Defense Method Based on Adaptive Simulated Annealing Genetic Algorithm*
(NAMBIAR et al., 2021)	*Multilevel Secure Container Deployment Framework in Edge Computing*
(ELSAYED; ZULKERNINE, 2019)	*Offering security diagnosis as a service for cloud SaaS applications*
(WALKER; LAIRD; CERNY, 2021)	*On Automatic Software Architecture Reconstruction of Microservice Applications*
(YIN; DU, 2020)	*On representing resilience requirements of microservice architecture systems*
(WASEEM et al., 2021)	*On the Nature of Issues in Five Open Source Microservices Systems: An Empirical Study*
(WONG et al., 2023)	*On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies*
(III; EL-SHEIKH, 2024)	*Open Platform Infrastructure for Industrial Control Systems Security*

Tabela 13 Continuação

Artigo	Título
(LIU et al., 2021)	*Operating Systems for Resource-adaptive Intelligent Software: Challenges and Opportunities*
(TOZER; MAZZUCHI; SARKANI, 2015)	*Optimizing Attack Surface and Configuration Diversity Using Multi-objective Reinforcement Learning*
(JIANG et al., 2023)	*OZTrust: An O-RAN Zero-Trust Security System*
(BHARDWAJ et al., 2021)	*Penetration testing framework for smart contract Blockchain*
(MAO et al., 2020)	*Preliminary Findings about DevSecOps from Grey Literature*
(KIM; KIM; LEE, 2021)	*Prof-gen: Practical Study on System Call Whitelist Generation for Container Attack Surface Reduction*
(WANG; KADIYALA; RUBIN, 2021)	*Promises and challenges of microservices: an exploratory study*
(EDDIN et al., 2024)	*Quantum Microservices: Transforming Software Architecture with Quantum Computing*
(CASALE et al., 2020)	*RADON: Rational Decomposition and Orchestration for Serverless Computing*
(LEE; CHOE; GHOSH, 2023)	*Recurrent Neural Network and Convolutional Neural Network for Detection of Denial of Service Attack in Microservices*
(CHEN et al., 2024)	*Research on the Security of Internet of Things Based on Microservices Techniques*
(WEN et al., 2023)	*Rise of the Planet of Serverless Computing: A Systematic Review*
(UNSEL et al., 2023)	*Risk-Based Authentication for OpenStack: A Fully Functional Implementation and Guiding Example*
(GIURA; JIM, 2018)	*Sapphire: using network gateways for IoT security*
(VERDERAME et al., 2023)	*SecCo: Automated Services to Secure Containers in the DevOps Paradigm*
(BUDIGIRI, 2023)	*Secure and Scalable Policy Management in Cloud Native Networking*
(OPPERMANN et al., 2018)	*Secure Cloud Computing : Risk Analysis for Secure Cloud Reference Architecture in Legal Metrology*

Tabela 13 Continuação

Artigo	Título
(NAM et al., 2023)	*Secure Inter-Container Communications Using XDP/eBPF*
(PAI; HEBBAR; KUMAR, 2023)	*Securing Microservice-Driven Applications Based on API Access Graphs Using Supervised Machine Learning Techniques*
(YANG et al., 2021)	*Security Challenges in the Container Cloud*
(KIM; PARK; SHIN, 2023)	*Security Consideration of Each Layers in a Cloud-Native Environment*
(THEODOROPOULOS et al., 2023)	*Security in Cloud-Native Services: A Survey*
(PEREIRA-VALE et al., 2021)	*Security in Microservice-Based Systems: A Multivocal Literature Review*
(WANG; WANG; HEMALATHA, 2024)	*Security Technology in Microservice Architecture*
(KLIMA et al., 2022)	*Selected Code-Quality Characteristics and Metrics for Internet of Things Systems*
(HATTORI; KATO, 2022)	*Sentinel: a fast and memory-efficient serverless architecture for lightweight applications*
(MCALEESE et al., 2022)	*Serverless Software Engineering – and How to Get There*
(MEADOWS et al., 2023)	*Sidecar-based Path-aware Security for Microservices*
(BéLAIR; LANIEPCE; MENAUD, 2021)	*SNAPPY: programmable kernel-level policies for containers*
(NKOMO; COETZEE, 2019b)	*Software Development Activities for Secure Microservices*
(SHEVCHUK et al., 2023b)	*Software for Improve the Security of Kubernetes-based CI/CD Pipeline*
(MINNA; MASSACCI, 2023)	*SoK: Run-time security for cloud microservices. Are we there yet?*
(BILLAWA et al., 2022)	*SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices*

Tabela 13 Continuação

Artigo	Título
(VASSILIOU-GIOLES, 2022)	*Solving the Instance Identification Problem in Micro-service Testing*
(CALDERóN-GóMEZ et al., 2020)	*Telemonitoring System for Infectious Disease Prediction in Elderly People Based on a Novel Microservice Architecture*
(RAMOS-CRUZ; ANDREU-PEREZ; MARTíNEZ, 2024)	*The cybersecurity mesh: A comprehensive survey of involved artificial intelligence methods, cryptographic protocols and challenges for future research*
(GOPAN et al., 2023)	*The Intercorrelation between Zero trust, Dark web, and its Implications in the field of Digital Forensics*
(LI et al., 2022)	*The Serverless Computing Survey: A Technical Primer for Design Architecture*
(PONCE et al., 2023)	*To Security and Beyond: On The Impacts of Microservice Security Smells and Refactorings*
(ALJAWAWDEH; SABRI; MAGHRABI, 2023)	*Toward Serverless and Microservices Architecture: Literature, Methods, and Best Practices*
(LI; CHEN; LIN, 2019)	*Towards Automated Inter-Service Authorization for Microservice Applicationss*
(KE; WU; YANG, 2021)	*Towards Evolving Security Requirements of Industrial Internet: A Layered Security Architecture Solution based on Data transfer Techniques*
(TOURANI et al., 2019)	*Towards security-as-a-service in multi-access edge*
(ALJAWAWDEH; ALJAIDI; MAGHRABI, 2024)	*Towards Serverless & Microservices Architecture: Strategies, Challenges, and Insights into Technology*
(ALABBAD; MHASKAR; KHEDRI, 2024)	*Two formal design solutions for the generalization of network segmentation*
(MINNA et al., 2021)	*Understanding the Security Implications of Kubernetes Networking*
(LYSNE et al., 2016)	*Vendor Malware: Detection Limits and Mitigation*

Tabela 13 Continuação

Artigo	**Título**
(KIM; KOO; KIM, 2022)	*Vulnerabilities and Secure Coding for Serverless Applications on Cloud Computing*
(CHEN, 2019)	*With Great Abstraction Comes Great Responsibility: Sealing the Microservices Attack Surface*
(SANKARAN; DATTA; BATES, 2020)	*Workflow Integration Alleviates Identity and Access Management in Serverless Computing*
(SHEN et al., 2019)	*X-Containers: Breaking Down Barriers to Improve Performance and Isolation of Cloud-Native Containers*
(DZOGOVIC et al., 2022)	*Zero-Trust Cybersecurity Approach for Dynamic 5G Network Slicing with Network Service Mesh and Segment-Routing over IPv6*
(XU et al., 2023a)	*Zero-trust security authentication based on spa and endogenous security architecture*