UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LORENA MAMEDE BOTELHO

ATRIBUIÇÃO DE MÉTRICA DE CONFIANÇA PARA INTERFACES DE PROGRAMAÇÃO EM ARQUITETURA DE MICROSSERVIÇOS

RIO DE JANEIRO
2024

LORENA MAMEDE BOTELHO

ATRIBUIÇÃO DE MÉTRICA DE CONFIANÇA PARA INTERFACES DE PROGRAMAÇÃO EM ARQUITETURA DE MICROSSERVIÇOS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadores: Prof. Evandro Luiz Cardoso Macedo,
Prof. Vinícius Gusmão Pereira de Sá

RIO DE JANEIRO
2024

CIP - Catalogação na Publicação

B748a

Botelho, Lorena Mamede
Atribuição de métrica de confiança para interfaces de programação em arquitetura de microsserviços / Lorena Mamede Botelho. -- Rio de Janeiro, 2024.
35 f.

Orientador: Vinícius Gusmão Pereira de Sá.
Coorientador: Evandro Luiz Cardoso Macedo.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2024.

1. Microsserviços. 2. Métrica de confiança. 3. Ataque de negação de serviço. 4. Análise estatística. I. Sá, Vinícius Gusmão Pereira de, orient. II. Macedo, Evandro Luiz Cardoso, coorient. III. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

LORENA MAMEDE BOTELHO

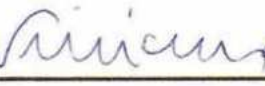
ATRIBUIÇÃO DE MÉTRICA DE CONFIANÇA PARA INTERFACES DE PROGRAMAÇÃO EM ARQUITETURA DE MICROSSERVIÇOS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 20 de agosto de 2024

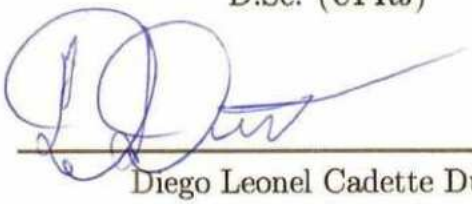
BANCA EXAMINADORA:

Evandro Luiz Cardoso Macedo
D.Sc. (UFRJ)



Vinícius Gusmão Pereira de Sá
D.Sc. (UFRJ)

Claudio Miceli de Farias
D.Sc. (UFRJ)



Diego Leonel Cadette Dutra
D.Sc. (UFRJ)

AGRADECIMENTOS

Este trabalho não seria possível sem a oportunidade e o apoio oferecidos pela Universidade Federal do Rio de Janeiro (UFRJ) e Instituto de Computação (IC). Agradeço por todo o aprendizado que me permitiu ser uma profissional e um ser humano melhor.

Aos meus orientadores, agradeço pelo voto de confiança em abraçar este projeto mesmo com todos os desafios que se impuseram, e por todas as contribuições que enriqueceram este trabalho. Ao prof. Evandro, em especial, agradeço pela mentoria, paciência e oportunidade de desenvolver esta proposta.

Agradeço à Equipe SIGA UFRJ, por ter me permitido durante todos esses anos de graduação ser parte da família SIGA, e gerado os frutos de boa parte dos conhecimentos que usei neste projeto, e que usarei no futuro. Em especial, agradeço ao coordenador da equipe, Ricardo Storino, por sempre acreditar em mim e dar a confiança necessária para a condução dos projetos que abracei. Aos meus parceiros de equipe, para os quais jamais poderia expressar a importância de tê-los junto comigo nesta jornada compartilhando todos os dias, de luta e de glória, da vida universitária.

Por fim, ao pilar de tudo que eu sou, minha família e amigos, por terem me escutado, incentivado, rido e chorado comigo durante toda essa trajetória. Ao meu namorado, Carlos Eduardo, pela paciência, pelas ideias e incentivo. E meu maior agradecimento à minha mãe, Patricia, por ser a minha maior orientadora, de profissão e de vida, cuja trajetória, conselhos, amor e experiência me permitiram chegar onde estou.

"First Principle: never to let one's self be beaten down by persons or by events"

Marie Curie

RESUMO

Este trabalho propõe uma metodologia para monitorar a comunicação entre microsserviços, com foco em interfaces de programação (*Application Programming Interface – API*) REST sob o protocolo HTTP. O modelo introduz dois indicadores: (i) a reputação do cliente, que classifica interações suspeitas com as APIs; e (ii) a saúde da aplicação, que permite identificar os *endpoints* sob ataques de DDoS e DoS, possibilitando a identificação de exploração de vulnerabilidades em tempo real. Para identificar ataques DDoS ou DoS, o modelo desenvolvido avalia a frequência de requisições usando uma distribuição de Poisson e divergência de Kullback-Leibler. A metodologia proposta combina análise estatística e medidas de segurança para monitorar microsserviços, oferecendo aos administradores dos serviços um panorama abrangente das possíveis ameaças e indicações de atuação. Através dos resultados obtidos, demonstramos a eficácia da abordagem em identificar atacantes e *endpoints* prejudicados.

Palavras-chave: microsserviços; métrica de confiança; ataque de negação de serviço; análise estatística.

ABSTRACT

This work proposes a methodology for monitoring communication between microservices, focusing on REST Application Programming Interfaces (API) under the HTTP protocol. The model introduces two indicators: (i) client reputation, which classifies suspicious interactions with APIs; and (ii) application health, which allows the identification of endpoints under DDoS/DoS attacks, enabling the real-time monitoring of vulnerability exploitation. To identify DDoS or DoS attacks, the developed model evaluates the frequency of requests using a Poisson distribution and Kullback-Leibler divergence. The proposed methodology combines statistical analysis and security measures to monitor microservices, offering service administrators a comprehensive overview of possible threats and indications for action. Through the results obtained, we demonstrate the effectiveness of the approach in identifying attackers and compromised endpoints.

Keywords: microservices; trust indicators; denial of service; statistical analysis.

LISTA DE ILUSTRAÇÕES

Figura 1 – Comparação entre Arquitetura Monolítica e Arquitetura de Microsserviços 13
Figura 2 – Métricas de taxa de tráfego em ataques DDoS para diferentes camadas de rede da Google Cloud 14
Figura 3 – Esquema da coleta de informações 22
Figura 4 – Variação das reputações iniciais atribuídas por C_1 27
Figura 5 – Distribuições com melhores resultados 28
Figura 6 – Distribuições com piores resultados 28
Figura 7 – Variação da taxa de requisição no tempo 29
Figura 8 – Variação da Métrica de Confiança no Cliente 29
Figura 9 – Variação da Métrica de Saúde do Enpoint 30

LISTA DE TABELAS

Tabela 1 – Análise qualitativa dos trabalhos relacionados 20
Tabela 2 – Parâmetros dos experimentos 27
Tabela 3 – Médias ponderadas das métricas de desempenho de cada distribuição . 29
Tabela 4 – Médias ponderadas das métricas de desempenho dos dados reais 30
Tabela 5 – Análise qualitativa dos trabalhos relacionados com a nossa proposta . . 31

LISTA DE ABREVIATURAS E SIGLAS

HTTP	Hypertext Transfer Protocol
DDoS	Distributed Denial-of-Dervice
DoS	Denial-of-Dervice
IP	Internet Protocol
SSRF	Server Side Request Forgery
API	Application Programming Interface
IoT	Internet of Things
TLC	Teorema do Limite Central
SOA	Service-Oriented Architecture
IDS	Intrusion Detection System

LISTA DE SÍMBOLOS

λ	parâmetro da distribuição Poisson
Pr	probabilidade
$\sigma(x)$	função sigmoide
$\tanh(x)$	função tangente hiperbólica

SUMÁRIO

1	**INTRODUÇÃO**	**12**
1.1	CONTEXTO E MOTIVAÇÃO	**12**
1.2	CONTRIBUIÇÕES DA METODOLOGIA	**16**
1.3	ORGANIZAÇÃO DO TRABALHO	**17**
2	**TRABALHOS RELACIONADOS**	**18**
2.1	DESCRIÇÃO DOS TRABALHOS RELACIONADOS	**18**
3	**METODOLOGIA**	**21**
3.1	COLETA DE DADOS E INFRAESTRUTURA PROPOSTA	**21**
3.2	MODELO DE CONFIANÇA PARA MICROSSERVIÇOS	**21**
3.2.1	**Cálculo de Confiança Inicial**	**22**
3.2.2	**Mapeamento do Perfil de Ataque DoS/DDoS**	**23**
3.2.3	**Atualização dos Indicadores**	**25**
4	**EXPERIMENTOS E VALIDAÇÃO DO MODELO**	**26**
4.1	TESTES COM DADOS SINTÉTICOS	**26**
4.1.1	**Análise da Componente C_1**	**26**
4.1.2	**Análise da Componente C_2**	**26**
4.2	SOBRE OS TESTES COM DADOS REAIS	**29**
5	**CONCLUSÃO**	**31**
5.1	CONTRIBUIÇÕES	**31**
5.2	DESAFIOS	**32**
5.3	TRABALHOS FUTUROS	**32**
	REFERÊNCIAS	**34**

1 INTRODUÇÃO

Este capítulo apresenta o contexto, problemas em aberto e motivação para o trabalho proposto. O texto é conduzido de modo a apresentar a relevância do tema, as contribuições e considerações no processo de desenvolvimento da solução abordada.

1.1 CONTEXTO E MOTIVAÇÃO

A Arquitetura de Microsserviços tem ganhado cada vez mais destaque na preferência dos desenvolvedores em alternativa à antiga Arquitetura Monolítica (Figura 1), dado que ela atende aos principais requisitos de entrega contínua (NEWMAN, 2021). Empresas como Netflix e Amazon já migraram da Arquitetura Monolítica para a de Microsserviços devido à rápida entrega e escalabilidade oferecidas. Cada vez mais empresas e setores adotam tal abordagem, inclusive setores que oferecem recursos essenciais para a população. Como exemplo, o governo brasileiro já tem usado esta alternativa em algumas das suas principais instituições como o Tribunal de Contas da União e o Ministério do Planejamento (LUZ et al., 2018). A grande vantagem na adoção de Microsserviços se dá pela heterogeneidade de tecnologias que podem ser usadas na construção de pequenas unidades autônomas de serviço, um fator relevante na escolha pela arquitetura em relação às aplicações monolíticas (LUZ et al., 2018). Todas as decisões de implementação são focadas no domínio da aplicação e, por isso, atendem melhor à variedade de regras de negócio de diferentes setores.

Detalhes de implementação ficam escondidos pelas *Application Programming Interfaces* (API), que oferecem um protocolo comum para a comunicação entre diferentes serviços. As APIs facilitam a troca de dados entre as aplicações, o que permite que uma gama de setores colete informações diversas, seja para mineração, cálculo de tendência, previsão de demandas, cadastro de usuários ou demais finalidades que ajudem a criar ferramentas inteligentes, automatizadas e adaptativas para o negócio. A adoção desta tecnologia por setores, cada vez mais diversos, exige uma alta disponibilidade dos serviços oferecidos, principalmente considerando os de grande importância para a sociedade, como sistemas de saúde, bancários, de ensino, entre outros. Quanto mais setores optam por esta arquitetura, teremos um volume cada vez maior de serviços presentes na rede, tornando a Arquitetura de Microsserviços gradualmente mais influente em nosso dia-a-dia.

A comunicação por APIs permite que os serviços sejam desacoplados uns dos outros. Como resultado, mudanças na implementação causam menos impacto nos demais, o que facilita o processo de entrega, sendo comum que uma aplicação seja dividida em múltiplos Microsserviços (MATEUS-COELHO; CRUZ-CUNHA; FERREIRA, 2021). Por isso, um número grande de *endpoints* é presente nesta arquitetura, o que, por sua vez, aumenta

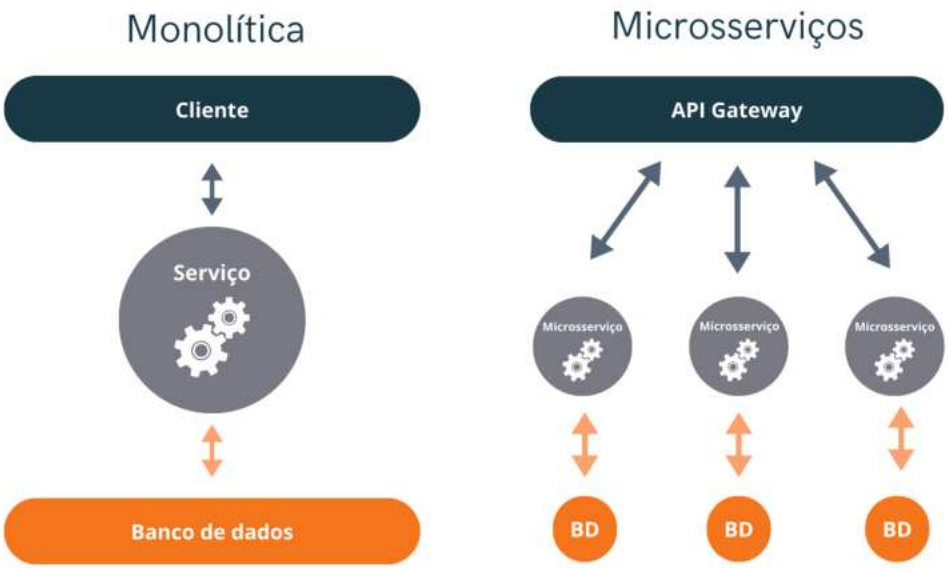


Figura 1 – Comparação entre Arquitetura Monolítica e Arquitetura de Microsserviços

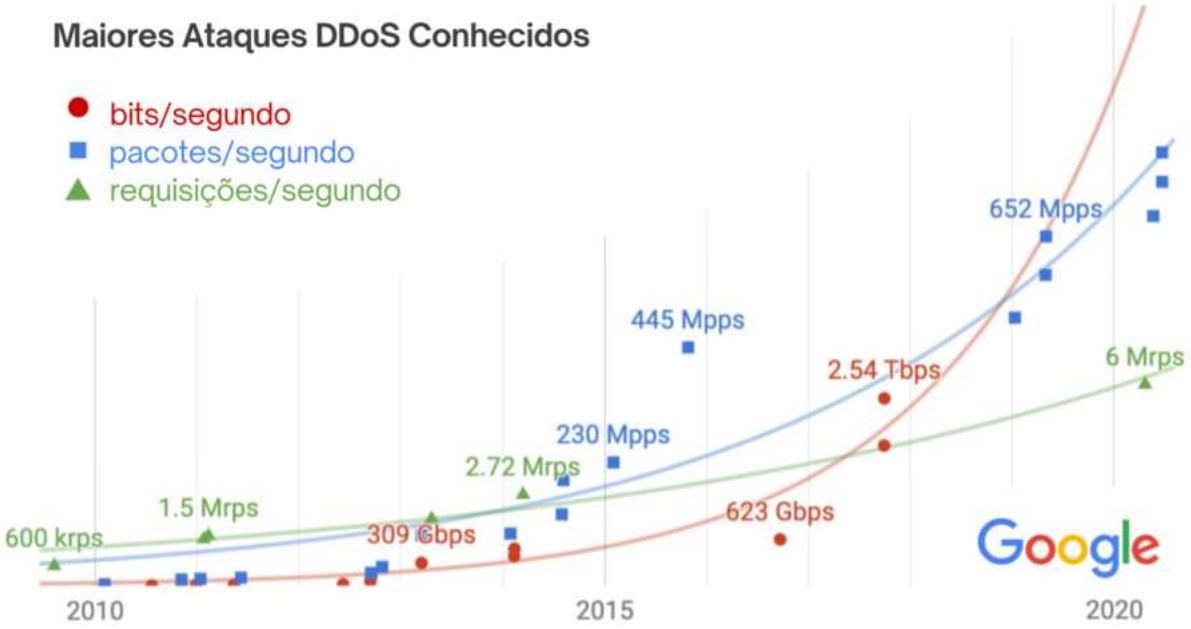
potencialmente a superfície de ataque, tornando ainda mais complexa a tarefa de monitoramento e detecção de possíveis ameaças (NEWMAN, 2021). Por exemplo, não é difícil que uma API que sofra poucas atualizações e receba menos atenção por parte dos desenvolvedores seja vulnerável e apresente brechas na exposição de dados, permitindo que um atacante consiga informações não autorizadas (SOWMYA et al., 2023). Outra possibilidade são ataques de *Denial of Service* (DoS), que buscam derrubar a aplicação ao realizar uma quantidade de requisições que supere a capacidade de atendimento do servidor, deixando os serviços fora do ar. Esse ataque também apresenta uma variação distribuída, o *Distributed Denial of Service* (DDoS), que possui o mesmo objetivo; a única diferença é que o ataque é realizado a partir de mais de um cliente, sendo distribuído em servidores ou *botnets*, que podem ser *hosts* que foram comprometidos em um ataque prévio que garantiu ganho de acesso indevido ao atacante. Não é surpresa que negação de serviço, além de ganho de acesso indevido e exposição de dados estejam entre os TOP 10 riscos da OWASP (OWASP, 2023).

Uma das características desta arquitetura também envolve a comunicação entre serviços. Como parte da divisão de responsabilidade, a aquisição de certos recursos é delegada a Microsserviços específicos aos quais outros realizam requisições para insumo de seus processos. Esta comunicação é, portanto, suscetível a ataques que se aproveitem da relação de segurança entre os Microsserviços, ocorrendo quando um atacante se apropria de um dos servidores do *cluster* permitindo se passar por um serviço lícito (MATEUS-COELHO; CRUZ-CUNHA; FERREIRA, 2021). Este ataque, *Server Side Request Forgery* (SSRF), é apenas um dos exemplos com os quais podemos ilustrar a necessidade de mecanismos

de avaliação constante das interações com os clientes requisitantes que se adapte ao dinamismo dessa comunicação.

Em um cenário em que a Arquitetura de Microsserviços se faz cada vez mais presente, as preocupações com segurança se tornam necessárias para garantir a proteção dos usuários que a utilizam. É fundamental que os desenvolvedores e a equipe de segurança tenham formas de monitorar em tempo real os *endpoints* para detectar e se recuperar mais rapidamente dessas falhas (CHANDRAMOULI, 2019).

As métricas de avaliação de ataques DDoS usadas pelo Google em seus serviços de *cloud* mostram o poder de mitigação que esta técnica apresenta. A Figura 2 mostra o monitoramento de taxa de bits por segundo(tráfego nos *links* de rede), taxa de pacotes (tráfego nos equipamentos de rede) e taxa de requisições (tráfego nos servidores de aplicações) dos diversos ataques DDoS sofridos pelo Google nos últimos tempos (MENSCHER, 2020). A análise de tendência permite que a empresa se planeje quanto a cargas futuras recebidas, e o monitoramento em tempo real permite uma resposta rápida aos eventos de ataque ou anomalia. Em seu relatório técnico, Menscher (MENSCHER, 2020) cita o uso de degradação graciosa e intervenções manuais como resposta a esses eventos, de forma que impeça a sua proliferação para diferentes camadas de rede. O relatório ressalta a margem de imprevisibilidade inerente aos ataques, o que dificulta prever seu comportamento. Essa preocupação mostra a relevância de se ter uma métrica que responda ao dinamismo em tempo real dos parâmetros de ataque durante a análise das comunicações com os serviços.



Fonte: Security and Identity Report 2020, Google Cloud

Figura 2 – Métricas de taxa de tráfego em ataques DDoS para diferentes camadas de rede da Google Cloud

O uso de métricas traz a vantagem de avaliar a evolução do cenário em direção a um estado. Isto permite não considerar apenas um único evento na sua classificação, e sim múltiplas ocorrências que levem à variação da métrica. Neste sentido, a métrica funciona como uma espécie de termômetro do problema, determinando intervalos que podem ser interpretados como diferentes estados de alerta. O desenvolvimento de métricas para confiança nas comunicações de uma rede é encontrado em diversos trabalhos, principalmente em aplicações para IoT (FORTINO et al., 2020) (MACEDO et al., 2022). Por outro lado, pouco foi desenvolvido para a área de Microsserviços, principalmente, na camada de aplicação. Os maiores desafios em se criar uma análise de confiança na Arquitetura de Microsserviços são lidar com sistemas abertos, que possuem comunicações externas às suas redes; lidar com problemas Cold Start, quando não há registro prévio da interação do cliente com o serviço, recorrente em sistemas Web, devido ao tempo de vida curto das sessões de comunicação; e como implementar esse controle na infraestrutura que é por definição composta por uma superfície grande de serviços (LU; DELANEY; LILLIS, 2023).

O trabalho de (LU; DELANEY; LILLIS, 2023) oferece a esta discussão a definição de quatro abordagens para modelos de confiança em Arquiteturas de Microsserviços: baseada em confiança-zero, baseada em sociedade, baseada em controle e baseada em composição. Na abordagem baseada em confiança-zero, é presumido que todas as interações podem ser hostis, portanto, há monitoramento constante sobre cada requisição que chega. Na baseada em sociedade, as interações e comportamentos passados entre Microsserviços são considerados na geração da métrica de confiança. Dentre as possíveis implementações para esta estratégia está a baseada em reputação, usada para computar a métrica como resultado da análise comportamental realizada. Na baseada em controle, é feito o uso de controle de acesso e controle de tráfego para restringir acesso e monitorar as interações. Na composição, os Microsserviços são avaliados em conjunto de acordo com a responsibilidade que compõem e, posteriormente, recebem uma avaliação indivual baseada na perfomance coletiva. As categorias desenvolvidas permitem nortear o desenvolvimento de novos modelos, inclusive combinando diferentes abordagens para elaboração de um modelo mais confiável.

Este trabalho busca explorar diferentes aspectos das abordagens anteriores, com preferência às abordagens baseadas em sociedade para criação de um sistema de reputação, às baseadas confiança-zero para análise de toda a requisição no *entrypoint* da arquitetura, e às baseadas em controle para monitoramento constante do tráfego de requisições. A partir dos trabalhos que se enquadram nestas categorias, esta pesquisa definiu alguns requisitos aproveitados por este trabalho: preferência por reputações recentes e monitoramento em tempo de execução do tráfego (AZARMI et al., 2012), (KRAVARI; BASSILIADES, 2019). Para atender aos requisitos de monitoramento, emerge o desafio de se automatizar a análise do tráfego, caracterizado pela problemática de se encontrar um cenário esperado para

comparar com o observado, dependendo da identificação de uma condição de normalidade que melhor expresse o comportamento ideal do sistema.

Neste contexto, o presente trabalho busca desenvolver métricas que auxiliem no monitoramento de aplicações que fazem uso de APIs, aproveitando as modelagens de métricas desenvolvidas em contextos de IoT, (MACEDO et al., 2022), adaptando-as para o cenário de Microsserviços. Com base na coleta dos dados das requisições, o trabalho propõe uma métrica para avaliar a confiança no cliente e outra para a saúde de aplicação, de modo a auxiliar em tomadas de decisão. Para o treinamento do tráfego esperado, utiliza análise estatística com o Teorema do Limite Central (TLC) para encontrar a melhor distribuição de Poisson que se ajuste ao cenário. Para monitoramento, o teste de qui-quadrado é utilizado para detecção da anomalia e a divergência de Kullback-Leibler (KULLBACK, 1959) para quantificar a variação na confiança. O trabalho descreve o processo de definição do perfil da aplicação durante ataques DoS e DDoS, para identificação de *endpoints* que estão sob ataque e dos clientes envolvidos. Além disso, também contribui com uma proposta de infraestrutura para coleta e processamento das informações que atenda às características da arquitetura. O objetivo é trazer um ferramental para a implementação de mais camadas de segurança na arquitetura de Microsserviços, incrementando o processo de rastreamento de ameaças ao oferecer alertas em tempo real para APIs. Os experimentos com diversas amostras sintéticas, simulando diferentes parâmetros de anomalia, corroboram a eficácia desta metodologia, abrindo portas para demais contribuições.

1.2 CONTRIBUIÇÕES DA METODOLOGIA

Este trabalho se diferencia por monitorar o tráfego entre aplicações, mais especificamente as requisições HTTP, enquanto outros trabalhos encontrados na literatura optaram por uma abordagem orientada a pacotes da camada de rede. Outra vantagem desta abordagem é a dependência apenas dos dados do cabeçalho das requisições HTTP, que podem ser facilmente aquisitados por diversos tipos de infraestrutura. O trabalho também oferece um esquema viável de implementação da metodologia em uma infraestrutura de Microsserviço qualquer. A abordagem inova ao trazer destaque para outro aspecto da segurança de Microsserviços, enquanto que nos demais trabalhos encontrados para este campo (MATEUS-COELHO; CRUZ-CUNHA; FERREIRA, 2021) (CHATTERJEE; PRINZ, 2022) a maior preocupação identificada foi em relação a métodos de controle de acesso, e não no monitoramento de tráfego de requisição para identificação de vulnerabilidades na comunicação cliente-serviço.

Muito já foi explorado por demais trabalhos (FORTINO et al., 2020)(MACEDO et al., 2022) com o uso de métricas de confiança no contexto de IoT. Esta proposta oferece uma transferência para o contexto de Microsserviços de todo o conhecimento adquirido na modelagem das técnicas anteriores. Portanto, apresenta a solidez dos trabalhos anteriores

e conduz aos primeiros passos para elaboração de futuros trabalhos acerca deste tema.

A escolha por uma abordagem estatística também é inovadora, já que diversos trabalhos voltados para detecção de ataques e anomalias utilizam técnicas de Aprendizado de Máquina (HINDY et al., 2020). Este trabalho mostra o grau de adaptabilidade que a análise estatística possui para diferentes cenários, além de promover uma visão mais clara do processo de classificação, cuja rastreabilidade dos padrões de classificação não costuma ser viável nas técnicas de Aprendizado de Máquina (ALANGARI et al., 2023). Nesse contexto, o presente trabalho também agrega com mais insumo para pesquisas que utilizem essa abordagem, oferecendo um novo objeto de estudo e de aplicação da técnica.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido da seguinte forma: o Capítulo 2 traz uma análise resumida de trabalhos relacionados. O Capítulo 3 descreve os parâmetros necessários para realização da modelagem matemática e detalha a infraestrutura projetada. O modelo é descrito através da elaboração de cada métrica envolvida e os cálculos necessários, o capítulo é encerrado com a formalização de todos os parâmetros para geração dos indicadores do cliente e da aplicação. O Capítulo 4 detalha os experimentos e resultados obtidos com o *dataset* gerado e discute o uso de dados reais. O Capítulo 5 conclui o trabalho, descreve suas limitações e sugere futuras pesquisas.

2 TRABALHOS RELACIONADOS

Através da análise de trabalhos anteriores, o objetivo deste capítulo é contextualizar o leitor quanto às lacunas identificadas na área de Segurança de Microsserviço, justificar a proposta deste trabalho e criar critérios para avaliação do cumprimento das expectativas deste projeto.

2.1 DESCRIÇÃO DOS TRABALHOS RELACIONADOS

A dificuldade em monitorar e proteger APIs na Arquitetura de Microsserviços está principalmente no grande número de Microsserviços colocado em produção, havendo uma quantidade significativa de *endpoints*. O volume de clientes interagindo com cada *endpoint* é ainda maior, podendo ser tanto usuários quanto outros serviços. Montar um perfil de ataque para esses clientes é complexo, pois envolve encontrar parâmetros adequados para identificá-los, endereços IPs podem ser falsificados ou mascarados e as conexões podem mudar constantemente. Há dificuldade em discernir um comportamento atípico de uma mudança legítima no tráfego da aplicação.

Mateus-Coelho *et al.* (MATEUS-COELHO; CRUZ-CUNHA; FERREIRA, 2021) descrevem os principais requisitos de segurança na implementação de Microsserviços, citando dentre muitos aspectos a proteção de APIs com uso de SSL/TLS, SAML, OpenID, API keys e HMAC dentre outros mecanismos de controle de acesso. Porém, a discussão é limitada apenas às alternativas de Autenticação e Autorização, não descrevendo métodos que permitam identificar comportamentos nocivos dos clientes no uso dos *endpoints*.

Chatterjee e Prinz (CHATTERJEE; PRINZ, 2022) buscaram, mediante um estudo de caso, criar uma implementação de controle de acesso para APIs, na Arquitetura de Microsserviços. O foco principal foi trazer o uso de plataformas de controle de acesso como o Keycloak, integrado ao *Spring Security Framework* e SAML para criar uma forma de autenticação e autorização mais segura. Uma infraestrutura digital foi apresentada para dificultar acesso não-autorizado ao *backend* da aplicação. O combate a DoS/DDoS e outros tipos de ataque foi delegado totalmente à implementação do *Spring Security Framework*, não sendo oferecido uma forma de monitoramento.

No trabalho (SOWMYA et al., 2023), os autores propõem o uso de *Random Forest* para monitoramento frequente de API considerando mensagens da camada de aplicação, com uso de compressor recursivo para extrair dados dos *logs*. *Pruned Exact Linear Time* é usado para detecção de ataques DoS. O problema em relação a usar os métodos propostos está principalmente no desafio de desempenho, além de ser condicionado a um balanceamento muito específico dos dados, não se adequando a qualquer ambiente de produção.

Os autores em (MACEDO et al., 2022) tratam da elaboração de uma métrica de confiança para avaliar a taxa de dados entre dispositivos IoT. O modelo combina a perspectiva de rede e de aplicação usando *blockchain* e técnicas de Teoria da Informação. Esta proposta faz uso das técnicas estatísticas e de Teoria da Informação descrita nesse trabalho, trazendo para o contexto de Microsserviços.

Em (ALJAZZAF; CAPRETZ; PERRY, 2016), os autores abordam a construção de uma métrica de confiança no contexto das Arquitetura Orientada a Serviço (*Service-Oriented Architecture* – SOA). Os autores apresentam um *framework* de confiança para serviços e provedores de serviço. A inicialização das métricas é feita por *Trust Bootstrapping*, e seu gerenciamento é feito por monitoramento. O cálculo das métricas de serviço é feito com parâmetros de tráfego de rede como latência e vazão, além de parâmetros de serviço como tempo de execução e tempo de resposta. O principal objetivo é criar métricas baseadas nas características de serviços e provedores de modo a permitir que a seleção por um cliente possa ser filtrada pelas métricas desejadas. O trabalho não foca em nenhum tipo de ataque, apenas oferecendo métodos para a melhor escolha de comunicação com serviços.

A maioria dos trabalhos de segurança na área de Microsserviços foca em configurações de autorização ou autenticação, além de abordarem soluções de monitoramento de pacotes da camada de rede, o que não é tão adequado quando se trata de monitorar a interação dos clientes com as APIs. Em alguns trabalhos, não é oferecida uma estratégia de monitoramento em tempo real. A elaboração de métricas que forneçam uma visão geral do estado da aplicação são aplicadas a outros cenários, que não o de Microsserviços. A Tabela 1 compara os trabalhos mencionados segundo cinco critérios de avaliação extraídos das lacunas identificadas nos trabalhos analisados: (i) Proteção de APIs, caso o trabalho tenha abordado diretamente a proteção de APIs; (ii) Aplicação para Microsserviços, classifica aqueles que se enquadram especificamente na área de Microsserviços; (iii) Métricas de Confiança, caso tenha abordado o uso ou elaboração de métricas de confiança; (iv) Monitoramento, categoriza propostas que desenvolveram métodos de monitoramento em tempo real de serviços ou dispositivos; e (v) Detecção de ataques DoS/DDoS, classifica técnicas desenvolvidas para detecção de ataques de negação de serviço. Este trabalho visa cobrir essas lacunas ao criar métricas que ajudem a identificar anomalias durante seu acontecimento, atendendo aos requisitos definidos pelas cinco categorias identificadas.

Tabela 1 – Análise qualitativa dos trabalhos relacionados

Trabalhos	Proteção de APIs	Aplicação para Micros-serviços	Métricas de Confiança	Monitoramento	Detecção de ataques DDoS/ DoS
Aljazzaf *et al.*, 2016			✓	✓	
Mateus-Coelho *et al.*, 2021	✓	✓			
Chatterjee e Prinz, 2022	✓	✓			
Macedo *et al.*, 2022			✓	✓	✓
Sowmya *et al.*, 2023	✓			✓	✓

3 METODOLOGIA

Neste capítulo, a infraestrutura de coleta e implementação da técnica é descrita. Em seguida, a modelagem matemática é desenvolvida em duas etapas iniciais: a modelagem da métrica de confiança inicial e a construção de um perfil de ataque DoS/DDoS. A seção final integra as duas modelagens para formalizar o cálculo dos indicadores de confiança do cliente e do *ednpoint*.

3.1 COLETA DE DADOS E INFRAESTRUTURA PROPOSTA

A interface mais comumente usada na comunicação de microsserviços são REST APIs, implementadas via protocolo HTTP. Este protocolo oferece uma série de métodos e parâmetros para a troca de informações na camada de aplicação. A análise usada neste trabalho consiste em usar dados dos cabeçalhos dessas requisições para mapear as características das entidades envolvidas na comunicação. Os seguintes dados da requisição são coletados: (i) o endereço IP do cliente que está tentando se comunicar com a API; (ii) o *endpoint* da API; e (iii) o *timestamp* da requisição.

Na Arquitetura de Microsserviço, a aplicação é fragmentada em múltiplas instâncias, usualmente via contêineres, cada uma contendo uma cópia do Microsserviço, que poderá expor múltiplos *endpoints*. Por isso, toda implementação da arquitetura conta com um tipo de *proxy* reverso para realizar a localização do serviço desejado e também direcionar a resposta para o cliente. Durante a comunicação do cliente com um *endpoint*, o cliente precisará passar por este *gateway* que será responsável por localizar o serviço requerido.

Aproveitando a característica desta infraestrutura, a coleta é projetada para ser realizada nesse ponto de contato, no qual um módulo responsável por extrair os dados da requisição é colocado, este módulo é chamado de Coletor de Requisições. A coleta é feita para cada requisição que passa pelo *gateway*. Um esquema ilustrando a infraestrutura proposta pode ser visualizado na Figura 3.

3.2 MODELO DE CONFIANÇA PARA MICROSSERVIÇOS

A fim de modelar o comportamento dos clientes e dos Microsserviços e traduzi-lo em uma métrica de confiança, o modelo oferece a criação de dois tipos de indicadores: a reputação do cliente (R_c) e a saúde da aplicação (R_a), com base no relatório de interação com os *endpoints* requisitados, ambos sendo definidos no intervalo entre 0.0 e 1.0. O perfil da aplicação é mapeado durante os ataques e as métricas que atualizam cada indicador são calculadas.

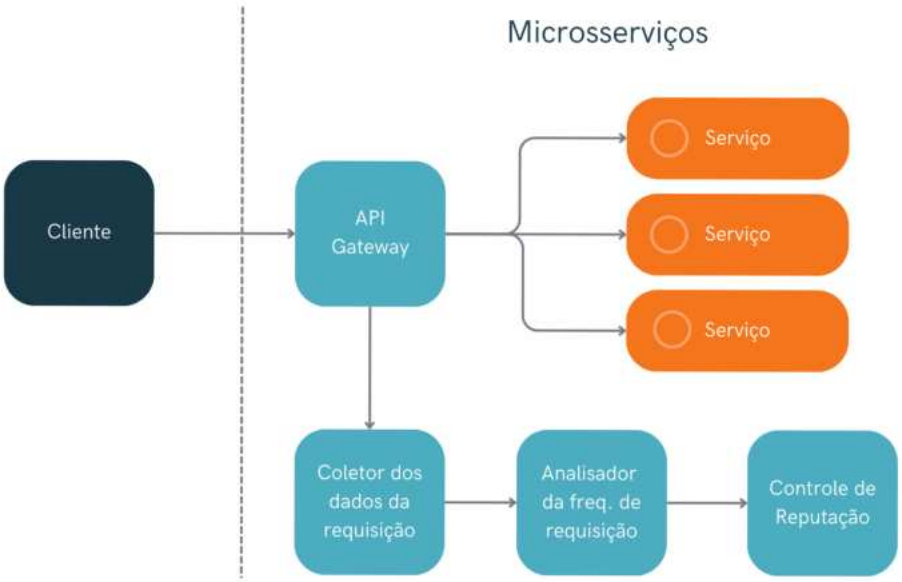


Figura 3 – Esquema da coleta de informações

3.2.1 Cálculo de Confiança Inicial

Para um *endpoint*, o estado inicial é configurado pelo próprio operador. Como regra geral, esse valor sempre será máximo (1.0), representando que o sistema está saudável.

O valor de reputação inicial (C_1) de um cliente expressa o valor inicial de confiança na primeira conexão com o serviço. É importante relembrar que na comunicação em termos de camada aplicação não é viável guardar um histórico do cliente a longo prazo, pois seus dados como endereço IP e *User Agent* podem ser facilmente modificados, por isso, a atualização da reputação é temporária e limitada ao tempo de sessão daquele IP associado. O início de uma sessão é definido pela primeira comunicação com o serviço, e toda a análise comportamental do cliente é a partir de sua sessão.

A fim de contornar o problema de Cold Start (PANDA; RAY, 2022), para o cálculo de C_1, o modelo desenvolve o conceito de grau de confiança do ecossistema, sendo basicamente o quão provável é um cliente, que interage com tal aplicação (ecossistema), agir maliciosamente. Serviços que têm um histórico de vulnerabilidade e passaram, ou estão passando, por ataques vão possuir maior probabilidade de que um novo cliente também seja malicioso. Esta métrica é usada para medir a saúde da aplicação de maneira geral e evitar que durante a exposição de uma vulnerabilidade, mais atacantes se aproveitem da brecha, o que pode contribuir na contenção de ataques DDoS.

Para estabelecer esse grau de confiança, o histórico de reputação dos clientes que já interagiram com o sistema, em um intervalo de tempo configurado pelo operador, é recuperado, dando prioridade para as reputações mais recentes. Deste intervalo, grupos de k reputações são extraídos de modo que seja possível obter um mínimo de 30 amostras.

Para cada amostra, a média amostral é calculada e, pelo Teorema do Limite Central, temos uma distribuição normal das médias. Seja m_i a reputação do i-ésimo cliente em uma amostra m de k clientes, o cálculo da média amostral é dado por:

$$\bar{M} = \frac{1}{k} \sum_{i=1}^{k} m_i \tag{3.1}$$

De posse desta distribuição, a probabilidade da reputação média dos clientes ser acima de γ (limiar de confiança alta, ajustável pelo operador) é calculada. Esta probabilidade é usada para gerar um valor aleatório no intervalo de β (limiar de confiança mínima, também justável pelo operador) e 1.0. Isso quer dizer que se a tendência de uma reputação alta for grande, o cliente terá maior probabilidade de iniciar com uma reputação alta. Sendo M a variável aleatória das médias amostrais, sendo $U(a, b)$ a distribuição uniforme, com a sendo o menor valor e b o maior valor do intervalo, o cálculo de C_1 é dado por:

$$C_1 = \begin{cases} U(\gamma, 1.0), \text{ se } U(0,1) < \Pr(M > \gamma) \\ U(\beta, \gamma - 0.01), \text{ caso contrário} \end{cases} \tag{3.2}$$

O cálculo dessa métrica fica dedicado ao módulo de Analisador de Frequência de Requisição, que fica responsável por analisar os dados que chegam ao Coletor de Requisições e identificar a chegada de novos clientes.

3.2.2 Mapeamento do Perfil de Ataque DoS/DDoS

Baseado na análise de ataques DoS e DDoS, o principal aspecto a ser considerado é a frequência de comunicação do cliente com o *endpoint*. O padrão desta frequência é ditado principalmente pelo intervalo entre as requisições. Por isso, o modelo usa a taxa de requisições por segundo feita a um determinado *endpoint*. A não diferenciação entre qual requisição é de qual cliente, nesta primeira análise, é importante, pois ataques distribuídos de negação de serviço podem ter vários clientes se revezando no disparo de requisições. Assim, o cálculo é feito sobre a quantidade geral de requisições que um *endpoint* está recebendo em um intervalo de tempo.

C_2 é a componente que avalia a frequência de requisições em um *endpoint*. Ela depende de um treinamento prévio para identificar a frequência usual. Assumindo as requisições como independentes, o treinamento é feito calculando a quantidade de requisições recebidas por *endpoint*, em cada intervalo de tempo t_i (em minutos) contido em um intervalo maior T, $t_i \in T, i \in [30, \infty)$. Para obter a taxa média de requisições por segundo, o total encontrado em cada t_i é dividido pelo seu total de segundos. Sendo L a variável aleatória das frequências por segundo encontradas, pelo Teorema do Limite Central, a distribuição d_L é normal e, portanto, a frequência de maior probabilidade é a sua média. A média λ estimada é usada como referência de frequência para analisar os futuros tráfegos observados.

Sendo x_j a quantidade de requisições no segundo j, durante o monitoramento, no mesmo intervalo de tamanho t_i usado na construção do modelo, é registrado o número x_j de requisições por instante de tempo, para $j \in [1, t_i * 60]$. Para cada x_j observado no intervalo t_i, a probabilidade Y_j esperada é calculada segundo uma distribuição $Poisson(\lambda)$, gerada com o λ estimado no treinamento. A escolha em usar esta distribuição para modelar o tráfego se deve pelo fato de ser apropriada para eventos independentes com ocorrências em intervalos de tempo, refletindo bem o comportamento das requisições. Além disso, a distribuição de Poisson só depende de um único parâmetro (λ), simplificando os cálculos.

Para avaliar se há divergência significativa entre o observado e o estimado, o teste qui-quadrado (χ^2) é usado. Este teste de hipótese se adequa bem a uma grande quantidade de classes, representadas pelos instantes de tempo discretizados na janela de tempo de observação; à independência entre as observações, característica das requisições; e à discretização dos dados analisados. Para este teste, a hipótese nula H_0 formulada é de que o tráfego observado, representado pela variável aleatória X de requisições por segundo, segue a distribuição Poisson com a média λ estimada, e H_1 é a hipótese contrária. A significância usada foi $\alpha = 0.05$, e os graus de liberdade $df = j-1$. Sendo O_j a probabilidade de x_j na distribuição d_X observada, temos:

$$\chi^2 = \sum_j \frac{(O_j - Y_j)^2}{Y_j} \tag{3.3}$$

Se $\chi^2 > \chi^2{}_{critico}$, então há anomalia no tráfego. Para atualizar o estado da aplicação, a divergência em relação ao tráfego estimado é extraída, e o valor de saúde da aplicação é decaído. A divergência usada foi Kullback-Leibler para obter a entropia relativa entre as distribuições.

$$D(d_E||d_X) = \sum_j d_{E_j}(x_j) log \frac{d_{E_j}(x_j)}{d_X(x_j)} \tag{3.4}$$

O D encontrado estará entre $[0, +\infty]$ e será usado para quantificar a redução ou aumento do indicador. Em caso de ser detectada a anomalia, a quantidade de requisições por cliente no *endpoint* prejudicado é verificada. Os clientes com alta quantidade de requisições para fazer D variar são penalizados juntamente com o *endpoint*.

Para calcular a componente, D é ajustado para ser realizada a atualização adequada do indicador. O limiar β é utilizado como critério de penalização, caso D seja menor que este valor (divergência baixa), a confiança deve aumentar, e caso D maior que β (divergência alta), a confiança deve diminuir. Ao decrescer D do limiar, temos valores não negativos para $D <= \beta$, e para $D > \beta$, valores negativos. Para limitar os valores negativos quando D é muito maior que 1.0 (divergência muito alta), a função de ativação tangente hiperbólica é escolhida para mapear o D encontrado entre -1 e 1. A escolha

desta função se deve pelo seu poder de limitar qualquer intervalo, os valores negativos são levado a $[-1, 0)$ e valores positivos a $(0, 1]$. Com isso, a componente ganha o poder de aumentar ou diminuir o valor dos indicadores sem extrapolar o intervalo destes. Isso é importante para a atualização dos indicadores, valores muito divergentes são penalizados e valores pouco divergentes permitem recuperação ao indicador.

$$C_2 = tanh(\beta - D) \tag{3.5}$$

Esse cálculo fica sob a responsabilidade do módulo Analisador de Frequência de Requisição, que verifica os dados que chegam no Coletor de Requisições em busca de frequências incomuns. Ao terminar o cálculo, a componente é enviada para o módulo de Controle de Reputação.

3.2.3 Atualização dos Indicadores

Atualização é realizada no módulo de Controle de Reputação, que recebe os relatórios enviados pelo Analisador de Frequência de Requisição. É importante ressaltar que no caso de clientes, a atualização é feita apenas em um intervalo de tempo em que se considera que o cliente analisado ainda pode ser identificado por determinado IP.

R_c é iniciado com C_1 e R_a inicia em 1 (ou outro valor arbitrário). À medida que as métricas são calculadas, $R_{c_{k+1}}$ e $R_{a_{k+1}}$ (próximos estados) são atualizados usando os valores calculados para C_2. Lembrando que C_2 apenas é computado para R_c em casos para os quais o cliente é identificado em um ataque de negação.

$$R_{c_{k+1}} = \sigma(R_{c_k} + C_2) \tag{3.6}$$

$$R_{a_{k+1}} = \sigma(R_{a_k} + C_2) \tag{3.7}$$

σ se refere à função sigmoide usada para que o valor de R continue limitado entre 0 e 1 após as atualizações.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.8}$$

Valores de R menores que β são considerados alertas sobre atividades maliciosas, e podem, em trabalhos futuros, implementarem ações temporárias, como adição em *block lists*.

4 EXPERIMENTOS E VALIDAÇÃO DO MODELO

Este capítulo descreve os testes implementados para validar o modelo, os dados e os parâmetros usados. O objetivo é verificar como o modelo se comporta em diferentes cenários, se efetivamente detecta anomalias e consegue classificar atacantes e *endpoints*.

4.1 TESTES COM DADOS SINTÉTICOS

Para estes testes, as amostras foram criadas com diferentes cenários, distribuições estatísticas e durações de anomalias. Os testes foram divididos em avaliações das atribuições de reputações iniciais da componente C_1, do poder de detecção de anomalia da componente C_2 e do seu poder de identificação dos atacantes e *endpoints* afetados.

4.1.1 Análise da Componente C_1

O cálculo de C_1 busca definir a reputação inicial do cliente baseado no grau de confiança do ecossistema, ou seja, nas reputações dos clientes que já estão interagindo com o sistema. Por conta disso, diferentes cenários foram criados com graus variados de anomalia, nos quais as reputações registradas refletem as seguintes situações: *com anomalia*, *em transição* e *sem anomalia*. Para cada um dos 3 casos, a componente deve pontuar um novo cliente entre β e 1.0, com probabilidades diferentes. Os testes foram executados fixando $\gamma = 0.8$ e $\beta = 0.5$.

Para o cenário *com anomalia*, as reputações iniciais estarão baixas, com a média em torno de 0.4, e a expectativa é de que novos clientes sejam classificados mais próximos de 0.5. Para o cenário *em transição*, as reputações estarão em torno de 0.6 (saindo de um cenário de anomalia), com a expectativa de que novos clientes sejam classificados mais próximos de 0.6. Para o cenário *sem anomalia*, as reputações estarão em torno de 0.8, logo, novos clientes devem ser classificados entre 0.8 e 1.0.

Como é possível ver nos gráficos obtidos na Figura 4, os resultados alcançados refletem o esperado, confirmando que C_1 reage bem às variações do ambiente. É importante ressaltar que, como o TLC é usado para estimar a reputação inicial, a quantidade de reputações anteriores usadas no cálculo é relevante. Para os testes, as últimas 1000 reputações foram utilizadas.

4.1.2 Análise da Componente C_2

A análise da componente C_2 permite testar tanto o poder de detecção quanto de responsabilização dos envolvidos. Para avaliar o poder de detecção, amostras com diferentes distribuições, durações e início de anomalia foram geradas. O interesse em testar diferen-

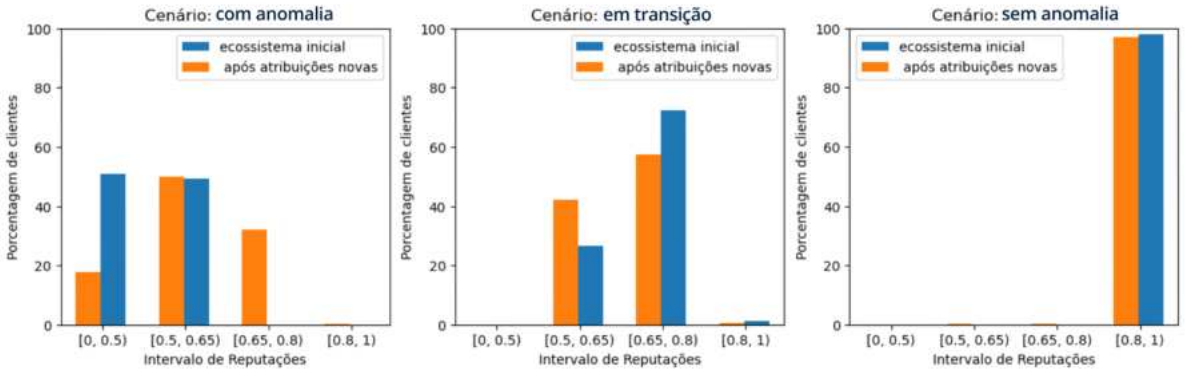


Figura 4 – Variação das reputações iniciais atribuídas por C_1

tes distribuições é verificar como o treinamento lida com variações na taxa de requisição que sejam muito discrepantes de uma distribuição de Poisson, usada pelo modelo para estimá-las. A Tabela 2 mostra os parâmetros escolhidos para geração das amostras em cada teste.

Tabela 2 – Parâmetros dos experimentos

Parâmetro	**Valor**
Cenários	Com anomalia e sem anomalia
Total de amostras	100 para cada cenário
Janela de tempo das amostras	300 segundos
Distribuições das amostras	Poisson, Binomial, Geométrica e Uniforme

As distribuições discretas foram escolhidas por se adequarem à discretização realizada nas observações utilizadas pelo método proposto. Como observado na Figura 5, os resultados demonstram uma boa acurácia nas predições para treinamento com tráfego observado em distribuição Poisson e Binomial. Isso pode ser explicado pela proximidade dessas duas distribuições. Além disso, o treinamento visa extrair da taxa de requisições observadas a Poisson que melhor descreve o tráfego. Portanto, quanto mais próxima a distribuição for de uma Poisson, mais acurada é a interpolação.

A Figura 6 mostra as distribuições com piores acurácias. Intuitivamente, a Uniforme é a que apresenta mais dificuldade em rastrear, dada sua alta variância, representando um fluxo onde qualquer taxa tem a mesma probabilidade de acontecer, não sendo possível determinar a discrepância entre fluxos anômalos e normais.

Os resultados mostram que o melhor cenário é aquele onde as taxas observadas seguem uma distribuição próxima ou igual a uma Poisson, isto favorece esta abordagem, pois dificilmente as requisições a uma API diferem desta distribuição. A Tabela 3 mostra as métricas de desempenho alcançadas por cada distribuição.

Para avaliar o poder de alerta dos indicadores, experimentos que simulam a variação

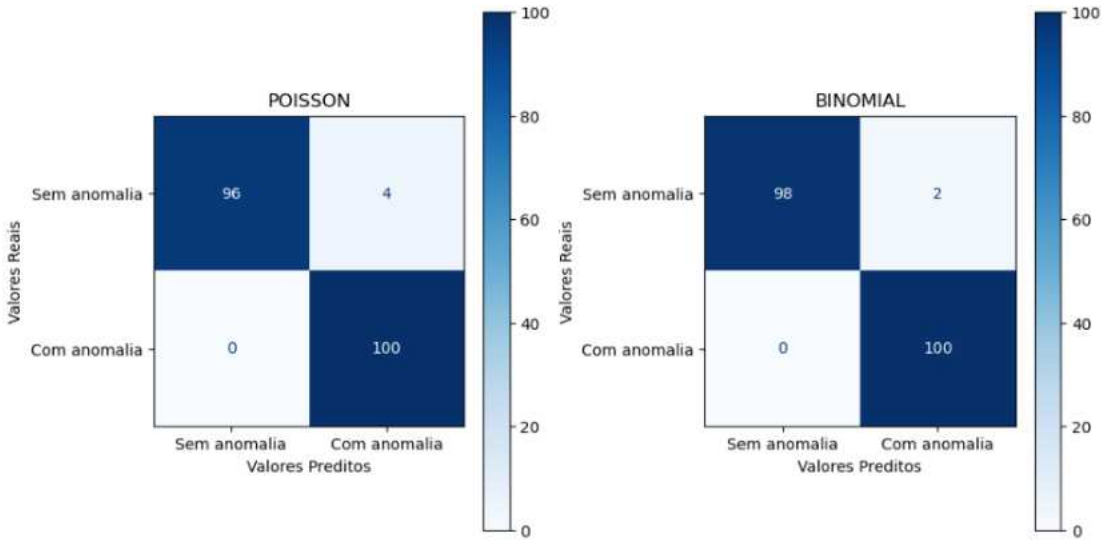


Figura 5 – Distribuições com melhores resultados

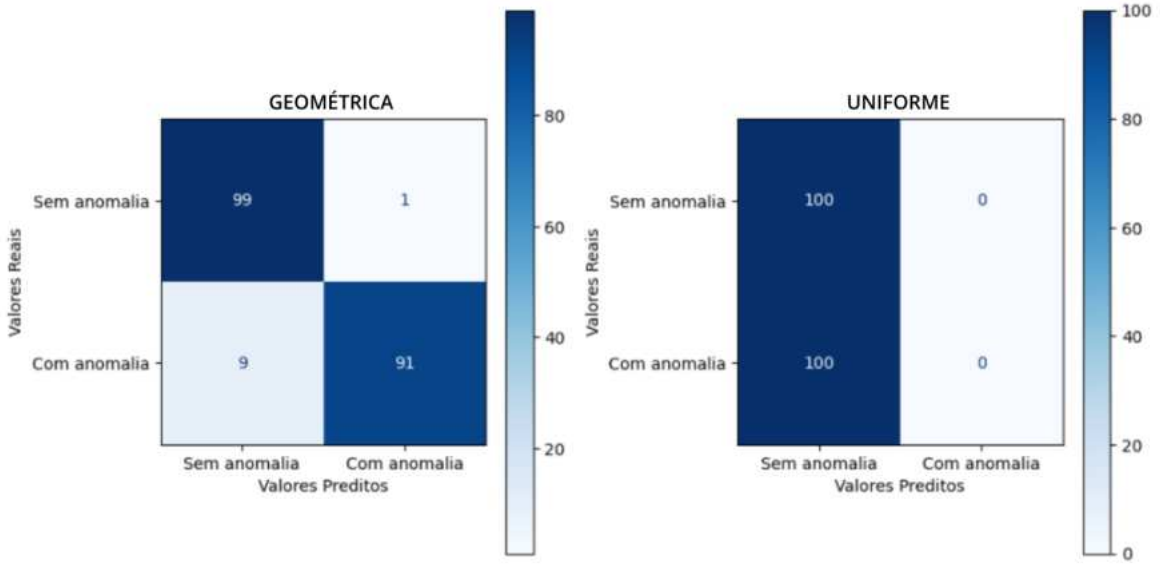


Figura 6 – Distribuições com piores resultados

dos indicadores para cliente e *endpoint* foram criados, com observação de 30 minutos. As anomalias foram geradas com tempos de duração, início/fim e intervalos diferentes. A intensidade foi aumentada gradualmente. Os parâmetros fixados foram: a distribuição em Poisson e a janela em 5 minutos. O experimento, ilustrado pela Figura 7, permite observar que após uma anomalia ser identificada na janela de tempo avaliada, há decaimento das métricas do cliente (Figura 8) e do *endpoint* (Figura 9). No ponto $1500s$, o gráfico permite notar que há recuperação, indicando que para intervalos muito pequenos, a janela não identificou anomalias. Quanto mais expressivos os intervalos e a intensidade, mais rápido o decaimento.

Tabela 3 – Médias ponderadas das métricas de desempenho de cada distribuição

Distribuição	**Acurácia**	**Precisão**	**Recall**	**F1-score**
Poisson	0.98	0.98	0.98	0.98
Binomial	0.99	0.99	0.99	0.99
Geométrica	0.95	0.95	0.95	0.95
Uniforme	0.50	0.25	0.50	0.33

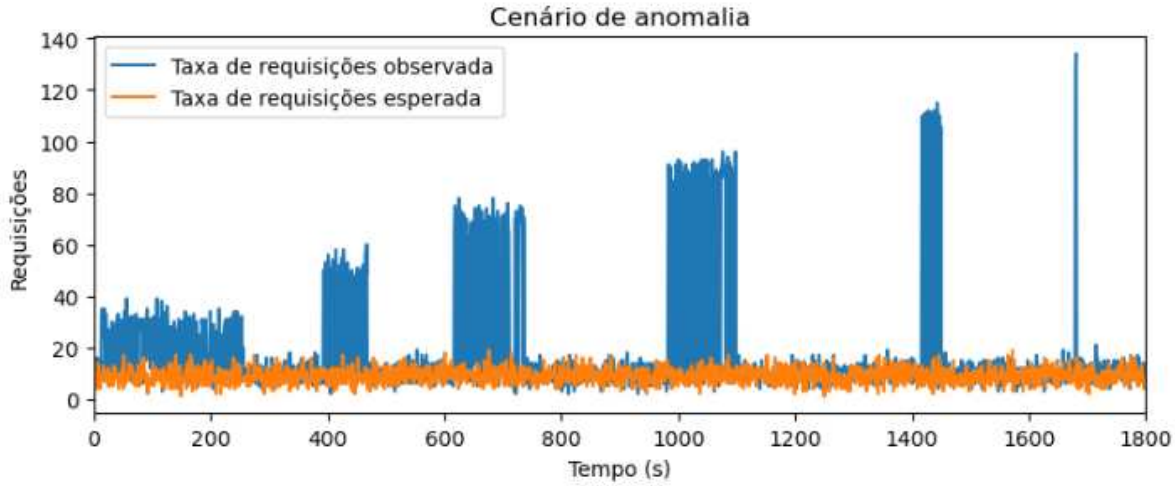


Figura 7 – Variação da taxa de requisição no tempo

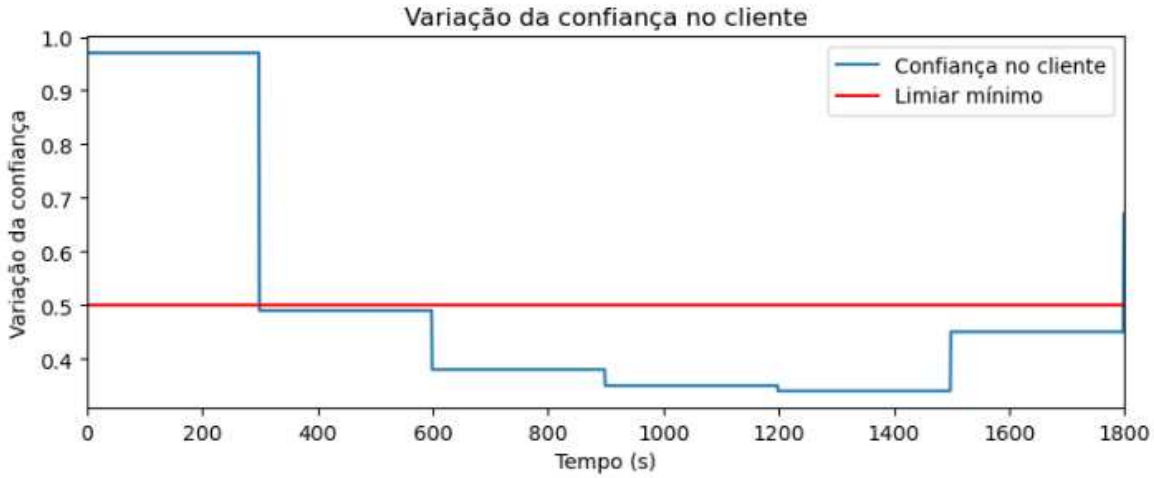


Figura 8 – Variação da Métrica de Confiança no Cliente

4.2 SOBRE OS TESTES COM DADOS REAIS

Como recomendado por Hindy *et al.*, (HINDY et al., 2020), para os testes com dados públicos, este trabalho utilizou o *dataset* disponibilizado pelo Canadian Institute for Cybersecurity (CIC), especificamente o CICDoS2019 (SHARAFALDIN et al., 2019). Apesar de ser reconhecido como o melhor *dataset* para treinamento de modelos de IDS atualmente, não apresentou a granularidade necessária para o problema estudado. Isto se

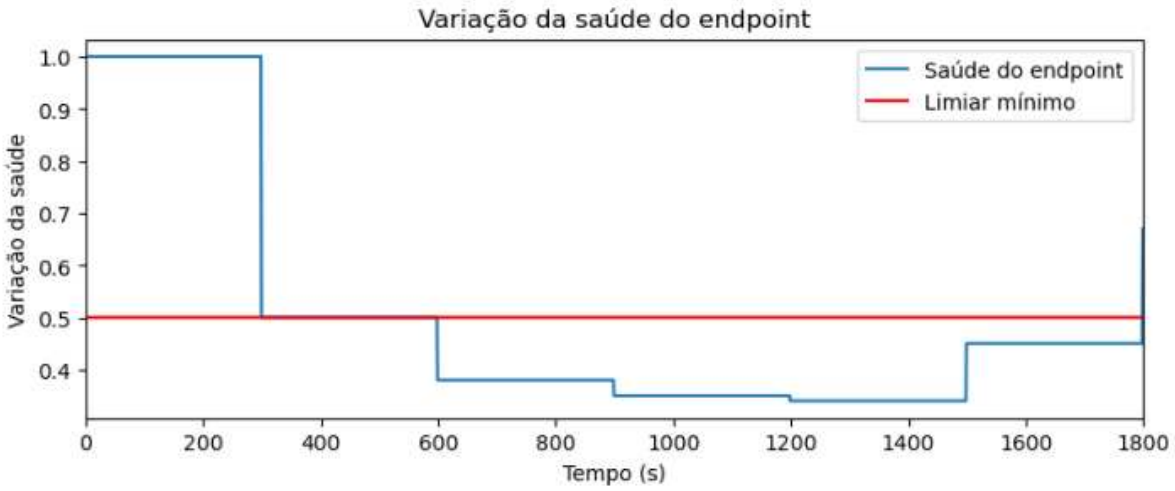


Figura 9 – Variação da Métrica de Saúde do Enpoint

deve pelo fato de a maioria dos *datasets* para IDS ser focada em técnicas de Aprendizado de Máquina. Segundo Hindy *et al.*, (HINDY et al., 2020), apenas 1% dos modelos de treinamento é estatístico, sendo 97% Aprendizado de Máquina e 2% Knowledge-based. Por este modelo desenvolvido ser estatístico, ele depende principalmente do *timestamp*, e de uma quantidade grande de intervalos de tempo para maior acurácia, que não é satisfatória nos *datasets* disponíveis atualmente.

Essa questão foi contornada com o uso de uma janela de tempo menor, de 30 segundos, para conseguir uma quantidade maior de janelas para treinamento. Obviamente, o modelo sofreu perda na sua acurácia. A Tabela 4 mostra os valores alcançados com 25 cenários de treinamento e, para testes, 5 com anomalia e 9 sem anomalia.

Tabela 4 – Médias ponderadas das métricas de desempenho dos dados reais

Métrica	**Valores**
Acurácia	0.50
Precisão	0.79
Recall	0.50
F1-score	0.44

5 CONCLUSÃO

5.1 CONTRIBUIÇÕES

Com este trabalho, apresentamos uma proposta para criação de métricas de confiança, que auxiliem no alerta e rastreio de ataques DoS/DDoS em APIs de Microsserviços. A automatização de alertas, com maior direcionamento para *endpoints* sob ataque e identificação de clientes nocivos, se mostra essencial na expansão do uso dos Microsserviços, principalmente com o aumento da escalabilidade desse tipo de sistema distribuído impulsionado pelo uso de contêineres.

A proposta é relevante ao contribuir com o estudo de uso de métricas para modelos de confiança no contexto da área de Segurança de Microsserviços, que possui literatura pouco amadurecida sobre o tema. A maioria dos estudos encontrados são aplicados a outros contextos, e aqueles que se aproximam mais da Arquitetura de Microsserviços, como SOA, já estão desatualizados. A tabela 5 mostra a contribuição deste trabalho em relação aos cinco requisitos que utlizamos para avaliar os trabalhos relacionados, demonstrando que esta nova abordagem supre uma lacuna não atendida pelos trabalhos anteriores.

Tabela 5 – Análise qualitativa dos trabalhos relacionados com a nossa proposta

Trabalhos	Proteção de APIs	Aplicação para Micros-serviços	Métricas de Confiança	Monitoramento	Detecção de ataques DDoS/ DoS
Aljazzaf *et al.*, 2016			✓	✓	
Mateus-Coelho *et al.*, 2021	✓	✓			
Chatterjee e Prinz, 2022	✓	✓			
Macedo *et al.*, 2022			✓	✓	✓
Sowmya *et al.*, 2023	✓			✓	✓
Esta proposta	✓	✓	✓	✓	✓

Os experimentos realizados mostraram a viabilidade da proposta e sua adaptabilidade ao comportamento dinâmico dos dados observados, abrindo portas para novos estudos que explorem o uso de análise estatística na modelagem de variáveis de confiança. Além disso, o trabalho apresenta uma proposta de implementação da solução em uma infraestrutura em conformidade com as características das Arquiteturas de Microsserviço. O trabalho

também agregou a contribuições anteriores com a combinação das categorias desenvolvidas por Lu *et al.* (LU; DELANEY; LILLIS, 2023), e ofereceu uma solução para problemas conhecidos na literatura como Cold Start (PANDA; RAY, 2022), ao abordar uma nova forma de inicialização de reputação para o primeiro registro de um cliente.

5.2 DESAFIOS

Apesar da solução proposta elevar o nível de segurança da aplicação, é preciso atentar-se aos desafios que ainda persistem. A possibilidade do IP poder ser falsificado adiciona uma camada a mais de complexidade, dificultando a identificação de clientes em ataques DDoS, por distribuir as taxas de requisição entre diferentes clientes, não permitindo atribuir uma maior participação a um único IP.

O uso de análise estatística requer o uso de um volume de dados grande para maior acurácia, podendo não ser a realidade de sistemas menores. A análise de requisições requer uma adaptação constante diante de um tráfego dinâmico, em caso de sistemas com variação constante da taxa de requisições, a mudança pode não acompanhar as rotinas de treinamento necessárias para o monitoramento acurado.

Destacamos também a dificuldade de realizar o treinamento com os *datasets* públicos disponíveis, devido à falta de intervalos de tempo longos o suficiente, e a ausência de tráfego limpo (sem anomalias), para realizar o treinamento mais efetivo.

5.3 TRABALHOS FUTUROS

O trabalho abre margem para que outras componentes C_i sejam adicionadas aos indicadores para fornecer alertas sobre outros tipos de ataque. É interessante explorar a combinação de diferentes modelagens matemáticas para a especificidade de cada componente que seja futuramente adicionada, já que não foram encontradas limitações que exigem homogeneidade no cálculo das mesmas. O uso de abordagens que permitam automatizar o treinamento podem ser interessantes, para permitir que o sistema aprenda de forma autônoma as características do seu fluxo.

Devido aos desafios com a testagem usando dados reais, este trabalho deixa como sugestão de futuras pesquisas a elaboração de *datasets* mais adequados para análises temporais e com diversidade de cenários de tráfego anômalo/benigno. O desenvolvimento deste tipo de dados pode auxiliar no fomento de abordagens diversificadas para modelos IDS.

No futuro, seria desejável realizar simulações em uma infraestrutura real para avaliar o desempenho e escalabilidade da solução. Uma possível abordagem encontrada em outros trabalhos, como o de Aljazzaf *et al.*, (ALJAZZAF; CAPRETZ; PERRY, 2016), explora uso de *brokers* e comunicações assíncronas para agendamento do cálculo das reputações.

Muito pode ser explorado em uma infraestrutura real, e acreditamos que possa ser de grande ganho para o desenvolvimento de futuros trabalhos.

REFERÊNCIAS

ALANGARI, N. et al. Exploring evaluation methods for interpretable machine learning: A survey. **Information**, v. 14, n. 8, 2023. ISSN 2078-2489. Disponível em: https://www.mdpi.com/2078-2489/14/8/469.

ALJAZZAF, Z. M.; CAPRETZ, M. A.; PERRY, M. Trust-based service-oriented architecture. **Journal of King Saud University - Computer and Information Sciences**, v. 28, n. 4, p. 470–480, 2016. ISSN 1319-1578. Disponível em: https://www.sciencedirect.com/science/article/pii/S1319157816300064.

AZARMI, M. et al. An end-to-end security auditing approach for service oriented architectures. In: **2012 IEEE 31st Symposium on Reliable Distributed Systems**. [S.l.: s.n.], 2012. p. 279–284.

CHANDRAMOULI, R. Microservices-based application systems. **NIST Special Publication**, v. 800, n. 204, p. 800–204, 2019.

CHATTERJEE, A.; PRINZ, A. Applying spring security framework with keycloak-based oauth2 to protect microservice architecture apis: A case study. **Sensors**, v. 22, n. 5, 2022. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/22/5/1703.

FORTINO, G. et al. Trust and reputation in the internet of things: State-of-the-art and research challenges. **IEEE Access**, v. 8, p. 60117–60125, 2020.

HINDY, H. et al. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. **IEEE Access**, v. 8, p. 104650–104675, 2020.

KRAVARI, K.; BASSILIADES, N. Storm: A social agent-based trust model for the internet of things adopting microservice architecture. **Simulation Modelling Practice and Theory**, v. 94, p. 286–302, 2019. ISSN 1569-190X. Disponível em: https://www.sciencedirect.com/science/article/pii/S1569190X19300322.

KULLBACK, S. **Information Theory and Statistics**. New York: Wiley, 1959.

LU, Z.; DELANEY, D. T.; LILLIS, D. A survey on microservices trust models for open systems. **IEEE Access**, v. 11, p. 28840–28855, 2023.

LUZ, W. et al. An experience report on the adoption of microservices in three brazilian government institutions. In: **Proceedings of the XXXII Brazilian Symposium on Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2018. (SBES '18), p. 32–41. ISBN 9781450365031. Disponível em: https://doi.org/10.1145/3266237.3266262.

MACEDO, E. L. C. et al. A two-level integrated approach for assigning trust metrics to internet of things devices. In: **International Conference on Internet of Things, Big Data and Security**. [s.n.], 2022. Disponível em: https://api.semanticscholar.org/CorpusID:248827129.

MATEUS-COELHO, N.; CRUZ-CUNHA, M.; FERREIRA, L. G. Security in microservices architectures. **Procedia Computer Science**, v. 181, p. 1225–1236, 2021. ISSN 1877-0509. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050921003719.

MENSCHER, D. **Exponential growth in DDoS attack volumes**. [S.l.], 2020. Disponível em: https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks.

NEWMAN, S. **Building Microservices**. 2nd. ed. [S.l.]: O'Reilly Media, Inc, 2021. ISBN 9781492034025.

OWASP. **OWASP Top 10 API Security Risks**. 2023. Https://owasp.org/API-Security/editions/2023/en/0x11-t10/.

PANDA, D. K.; RAY, S. Approaches and algorithms to mitigate cold start problems in recommender systems: a systematic literature review. **Journal of Intelligent Information Systems**, v. 59, 2022. ISSN 1573-7675.

SHARAFALDIN, I. et al. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: **2019 International Carnahan Conference on Security Technology (ICCST)**. [S.l.: s.n.], 2019. p. 1–8.

SOWMYA, M. et al. Api traffic anomaly detection in microservice architecture. p. 206–213, 2023.