Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de informação

Arquitetura Zero Trust como Microsserviço Serverless em Nuvem com ABAC e Machine Learning

Rodrigo Soares de Assis

JUIZ DE FORA
JANEIRO, 2026

Arquitetura Zero Trust como Microsserviço Serverless em Nuvem com ABAC e Machine Learning

Rodrigo Soares de Assis

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Sistemas de informação

Orientador: Alex Borges Vieira

JUIZ DE FORA

JANEIRO, 2026

Arquitetura Zero Trust como Microsserviço Serverless em Nuvem com ABAC e Machine Learning

Rodrigo Soares de Assis

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Alex Borges Vieira
Doutorado em Ciência da Computação

Luciano Jerez Chaves
Doutorado em Ciência da Computação

Edelberto Franco Silva
Doutorado em Computação

JUIZ DE FORA

21 DE JANEIRO, 2026

À minha família, que sempre me deu o suporte necessário, especialmente à minha mãe, que sempre batalhou e fez de tudo para oferecer o melhor aos filhos, incluindo a minha educação.

Resumo

A crescente adoção de computação em nuvem impõe novos desafios à segurança da informação, especialmente no controle de acesso em ambientes distribuídos. Nesse contexto, a Arquitetura Zero Trust propõe a eliminação da confiança implícita, exigindo a avaliação contínua do contexto de cada requisição. Este trabalho apresenta uma arquitetura de decisão de acesso baseada nos princípios do Zero Trust, implementada como uma API serverless em nuvem. A solução combina políticas determinísticas de controle de acesso baseadas em atributos (ABAC) com um modelo de aprendizado de máquina supervisionado, utilizado como mecanismo complementar de estimativa de risco contextual. A metodologia envolve uma revisão sistemática da literatura, conduzida segundo PRISMA e PICOC, seguida do desenvolvimento de uma prova de conceito. Os resultados indicam que a abordagem proposta é viável e funcional, permitindo decisões de acesso mais sensíveis ao contexto. São discutidas limitações relacionadas ao uso de dados sintéticos e ao tratamento conceitual da autenticação multifator, bem como direções para trabalhos futuros.

Palavras-chave: Arquitetura Zero Trust, Controle de Acesso Baseado em Atributos, Autorização Baseada em Risco, Aprendizado de Máquina, Segurança em Nuvem.

Abstract

The increasing adoption of cloud computing environments introduces new challenges to information security, particularly regarding access control in distributed systems. In this context, the Zero Trust Architecture eliminates implicit trust by requiring continuous evaluation of the context of each access request. This work presents an access decision architecture based on Zero Trust principles, implemented as a serverless API in a cloud environment. The proposed solution combines deterministic attribute-based access control (ABAC) policies with a supervised machine learning model, used as a complementary mechanism for contextual risk estimation. The methodology includes a systematic literature review conducted according to PRISMA and PICOC guidelines, followed by the development of a proof of concept. The results indicate that the proposed approach is viable and functional, enabling more context-aware access decisions. Limitations related to the use of synthetic data and the conceptual treatment of multi-factor authentication are discussed, as well as directions for future work.

Keywords: Zero Trust Architecture, Attribute-Based Access Control, Risk-Based Authorization, Machine Learning, Cloud Security.

Agradecimentos

Agradeço à minha família, em especial aos meus pais, por todo o suporte, incentivo e compreensão ao longo desses quase cinco anos. Sem eles, nada disso teria sido possível.

Sou especialmente grato à minha mãe, Fabiana Soares de Melo Assis, que nunca deixou de acreditar em mim nem permitiu que eu desacreditasse de mim mesmo. Ela jamais mediu esforços, muitas vezes abrindo mão do próprio conforto, para garantir que eu tivesse acesso à educação e às oportunidades que me trouxeram até aqui.

Aos meus amigos, que tornaram esses anos longe da minha família mais leves e acolhedores, e com quem vivi momentos que levarei para sempre comigo.

Ao Diogo, que hoje considero como um irmão, pelo apoio constante em todos os momentos e por estar ao meu lado desde o início dessa caminhada.

Ao professor Alex Borges Vieira, do Departamento de Ciência da Computação, pela orientação neste trabalho e por me apresentar o mundo das Redes de Computadores, área que hoje faz parte do meu dia a dia profissional e da trajetória que escolhi seguir.

Aos professores do Departamento de Ciência da Computação, pelos ensinamentos transmitidos ao longo da graduação, e aos funcionários do curso, que de diferentes formas contribuíram para o meu crescimento pessoal e profissional.

"Everything you lose is a step you take...

You've got no reason to be afraid."

Taylor Swift — You're On Your Own, Kid

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computução
UFJF	Universidade Federal de Juiz de Fora
ABAC	*Attribute-Based Access Control*
RBAC	*Role-Based Access Control*
ZTA	*Zero Trust Architecture*
mTLS	*Mutual Transport Layer Security*
JWT	*JSON Web Token*
PRISMA	*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*
PICOC	*Population, Intervention, Comparison, Outcome, Context*
PEP	*Policy Enforcement Point*
PDP	*Policy Decision Point*
MFA	*Multi-Factor Authentication*
NIST	*National Institute of Standards and Technology*
AWS	*Amazon Web Services*
IaC	*Infrastructure as Code*

1 Introdução

Com o desenvolvimento de empresas e economias digitais, os dados e as redes se tornaram elementos indispensáveis. A crescente digitalização trouxe inúmeras conveniências, mas também resultou em um aumento expressivo nas ameaças à segurança da informação, especialmente dentro das próprias redes corporativas (KANG et al., 2023).

A proteção da informação consolidou-se como um dos principais desafios da era digital, tanto pelo seu impacto direto na continuidade dos negócios quanto pelos prejuízos financeiros associados a falhas de proteção. Violações de dados, sequestros de informações e ataques direcionados causam bilhões de doláres em perdas todos os anos, afetando desde pequenas empresas até grandes corporações e instituições governamentais. Somente no setor financeiro, bancos, seguradoras e gestoras de ativos sofreram mais de 20 mil ataques cibernéticos nas últimas décadas, gerando perdas de aproximadamente US$ 12 bilhões ao setor, segundo relatório do Fundo Monetário Internacional (FMI) (EXAME, 2024).

Tradicionalmente, a segurança de rede era baseada no conceito de perímetro de segurança, no qual a infraestrutura era dividida entre uma rede interna, considerada confiável, e uma rede externa, considerada não confiável (NORTHCUTT et al., 2005). No entanto, esse modelo mostra-se limitado diante das ameaças internas, pois foca em uma defesa unidirecional, tornando-se ineficaz contra ataques originados dentro do próprio ambiente corporativo.

Nesse contexto, surge o conceito de Zero Trust, um novo paradigma de segurança cibernética que rompe com o modelo tradicional baseado em perímetros. Esse modelo parte do princípio de "nunca confie, sempre verifique", ou seja, todas as requisições, internas ou externas, devem ser autenticadas, validadas e autorizadas antes de acessar recursos da rede. Esse modelo se baseia em três pilares principais: (1) todas as fontes devem ser verificadas e protegidas; (2) o controle de acesso deve ser restrito e rigorosamente aplicado; e (3) todo o tráfego de rede deve ser monitorado, inspecionado e registrado (KINDERVAG, 2010).

Diante desse cenário, este trabalho propõe uma arquitetura de decisão de acesso

baseada nos princípios da Arquitetura Zero Trust, implementada como um microsserviço *serverless* em ambiente de nuvem. A proposta combina políticas determinísticas de controle de acesso baseadas em atributos (ABAC) com um modelo de aprendizado de máquina supervisionado, utilizado como estimador de risco contextual no processo de autorização. Essa abordagem busca oferecer maior flexibilidade e adaptabilidade às decisões de acesso, mantendo previsibilidade, explicabilidade e aderência aos princípios de verificação contínua do modelo Zero Trust.

A avaliação da arquitetura proposta foi conduzida por meio de uma prova de conceito, na qual foram analisados diferentes cenários de requisição de acesso com variações nos atributos de usuário, recurso e contexto. Os resultados indicaram que a integração entre políticas ABAC e um modelo de aprendizado de máquina como estimador de risco é viável e permite decisões de acesso mais sensíveis ao contexto, preservando a previsibilidade das regras determinísticas. Além disso, os experimentos evidenciaram desafios relacionados ao uso de dados sintéticos e à formulação do cálculo de risco, os quais motivaram ajustes no modelo e contribuíram para uma melhor compreensão do papel complementar do aprendizado de máquina em arquiteturas Zero Trust.

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta os trabalhos relacionados, fundamentados em um mapeamento sistemático da literatura sobre Zero Trust em ambientes de microsserviços e computação em nuvem. O Capítulo 3 descreve a metodologia adotada, incluindo o desenvolvimento da prova de conceito. O Capítulo 4 apresenta os resultados obtidos na avaliação experimental da arquitetura proposta. Por fim, o Capítulo 5 discute as conclusões e aponta direções para trabalhos futuros.

2 Trabalhos Relacionados

Os trabalhos relacionados apresentados neste capítulo foram selecionados a partir de uma revisão sistemática da literatura conduzida conforme as diretrizes PRISMA (PRISMA Group, 2020) e estruturada pelo método PICOC (WOHLIN et al., 2012). A revisão teve como foco a identificação de estudos que abordassem mecanismos de segurança alinhados aos princípios da ZTA em ambientes de microsserviços e computação em nuvem, a partir de buscas realizadas nas bases IEEE Xplore, Scopus e ACM Digital Library.

Nesse contexto, a literatura recente tem investigado a aplicação dos princípios da Arquitetura Zero Trust em ambientes baseados em microsserviços e computação em nuvem, motivada pelo aumento da superfície de ataque e pela complexidade inerente aos sistemas distribuídos.

Diversos trabalhos concentram-se na aplicação do modelo Zero Trust no plano de comunicação entre microsserviços. Em (MILLER et al., 2021), os autores propõem o isolamento de microsserviços por meio de mecanismos de controle acoplados à arquitetura, reforçando princípios como segmentação e verificação contínua. De forma semelhante, (VISWANATHAN; N; KUMAR, 2024) discute a aplicação do Zero Trust em aplicações web baseadas em microsserviços, destacando o uso de autenticação baseada em tokens JWT, comunicação segura via mTLS e políticas de controle de acesso aplicadas de forma centralizada.

Outro conjunto relevante de trabalhos explora o uso de *service mesh* como base para a implementação de arquiteturas Zero Trust. Estudos como (POUDEL et al., 2025) e (MONTEBUGNOLI; SABBIONI; FOSCHINI, 2024) utilizam tecnologias como Istio e Envoy para interceptar o tráfego entre serviços, aplicar políticas de segurança e garantir autenticação mútua. Essas abordagens demonstram a viabilidade do Zero Trust em arquiteturas altamente distribuídas, porém concentram-se predominantemente no plano de controle e na comunicação serviço-a-serviço, com menor ênfase na decisão de acesso no ponto de entrada da aplicação.

No que se refere aos mecanismos de autenticação e autorização, observa-se a

adoção recorrente de modelos tradicionais, como o *Role-Based Access Contro*l (RBAC) e o *Attribute-Based Access Control* (ABAC), combinados com identidades baseadas em certificados digitais e tokens. O trabalho apresentado em (FALCãO et al., 2022) destaca o uso do SPIRE e do padrão SPIFFE para emissão de identidades de workloads em ambientes de computação confidencial. Já em (KHOLIDY et al., 2023), é proposta uma extensão do modelo RBAC tradicional, denominada TRBAC, que incorpora elementos de confiança dinâmica para reavaliação contínua das permissões de acesso em arquiteturas nativas de nuvem.

Apesar desses avanços, poucos trabalhos exploram de forma aprofundada o uso de aprendizado de máquina como suporte direto ao processo de decisão de acesso. O estudo apresentado em (ALBOQMI; JAHAN; GAMBLE, 2023) propõe um mecanismo de avaliação de confiança em tempo de execução baseado em dados telemétricos coletados no *service mesh*, mas não integra explicitamente modelos de aprendizado supervisionado nem aborda o controle de acesso contextual no limite entre usuários e sistemas.

De modo geral, observa-se que a maioria das abordagens existentes prioriza a proteção da comunicação interna entre microsserviços, dedicando menor atenção à decisão de acesso contextual baseada em atributos de usuários, recursos e ambiente. Além disso, observa-se que poucos trabalhos exploram a combinação entre políticas determinísticas baseadas em atributos e modelos de aprendizado de máquina de forma modular e explicável, especialmente no contexto da decisão de acesso.

Nesse contexto, o presente trabalho diferencia-se ao propor uma arquitetura de decisão de acesso baseada em Zero Trust, implementada como um microsserviço serverless, que integra políticas ABAC com um modelo de aprendizado de máquina supervisionado para estimativa de risco contextual. Essa abordagem busca preencher lacunas identificadas na literatura, oferecendo uma solução mais flexível, adaptativa e alinhada às demandas de ambientes modernos em nuvem.

3 Metodologia

A metodologia deste trabalho foi estruturada em duas etapas principais: (i) uma revisão sistemática da literatura, que fundamenta o estudo teórico sobre o modelo Zero Trust em ambientes de microsserviços e nuvem; e (ii) o desenvolvimento de uma prova de conceito (PoC) implementada parcialmente em ambiente AWS, com o objetivo de avaliar a aplicabilidade prática das abordagens encontradas.

3.1 Prova de Conceito (PoC)

Com base nas evidências obtidas na revisão da literatura, foi desenvolvida uma prova de conceito representando um ambiente simulado em nuvem, composto por uma API serverless responsável pela decisão de acesso, estruturada de forma modular e alinhada aos princípios da arquitetura Zero Trust. A solução implementa um mecanismo de decisão baseado em políticas ABAC, complementado por um modelo de aprendizado de máquina para estimativa de risco.

A prova de conceito foi implementada inicialmente em ambiente local por meio de uma API baseada em HTTP, com o objetivo de facilitar o desenvolvimento, a validação funcional e a análise dos mecanismos de decisão. Posteriormente, a arquitetura foi implantada em ambiente de nuvem, visando demonstrar a viabilidade da solução em um contexto serverless e distribuído.

3.2 Arquitetura Experimental

A PoC foi projetada para reproduzir um ambiente corporativo heterogêneo, composto por diferentes serviços e usuários com distintos níveis de privilégio, a fim de avaliar a aplicação de políticas de segurança dinâmicas baseadas em Zero Trust.

A arquitetura foi dividida nos seguintes módulos principais:

- Policy Decision Point (PDP): módulo central de decisão, responsável por aplicar as

regras de controle de acesso baseadas em atributos e por integrar um modelo de aprendizado de máquina supervisionado utilizado como estimador de risco, a partir do qual são definidas as decisões de acesso *allow, challenge ou deny*.

- Policy Enforcement Point (PEP): componente que intercepta as requisições enviadas e aplica a decisão de acesso retornada pelo PDP. Além de permitir ou negar operações, o PEP também pode redirecionar a requisição para um fluxo de verificação adicional (*challenge*), assegurando que as políticas de Zero Trust sejam aplicadas de forma consistente.

A requisição passa inicialmente pela validação das políticas ABAC. Para ser aprovada nessa etapa, deve existir ao menos uma política que autorize o acesso e nenhuma que o negue; caso contrário, a requisição é automaticamente rejeitada.

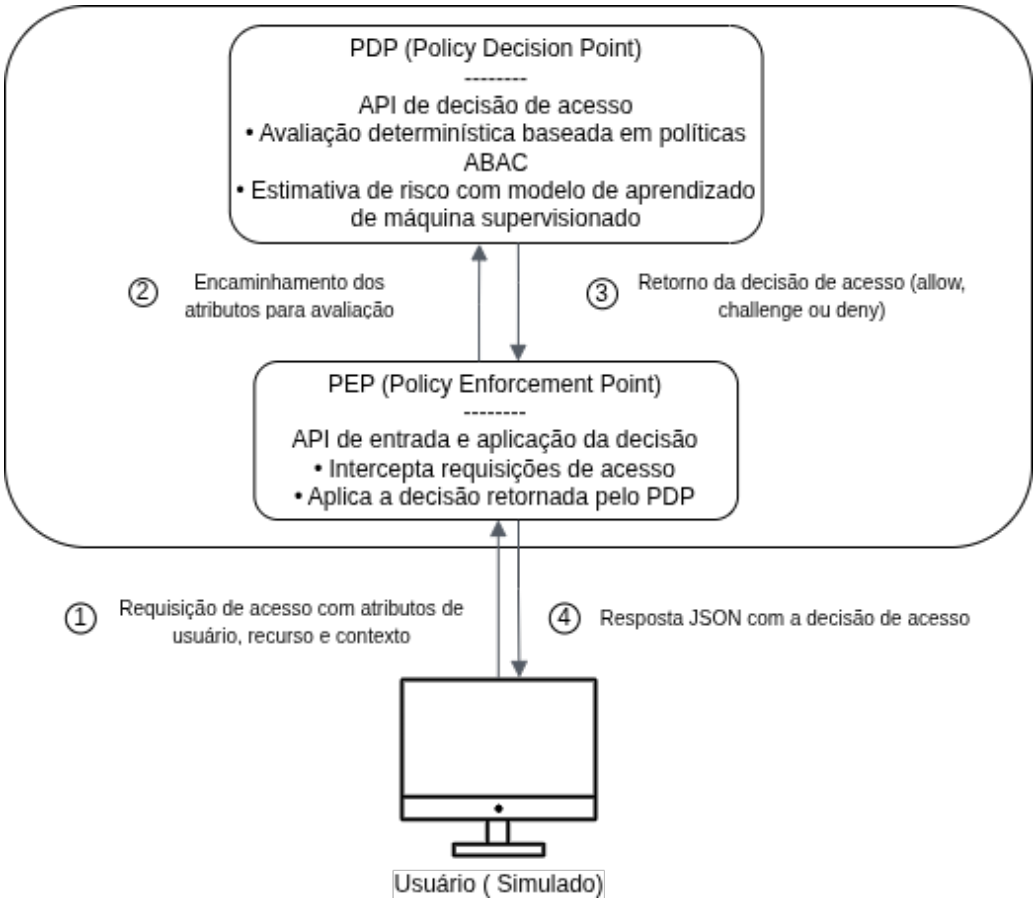

Figura 3.1: Arquitetura experimental ilustrando o fluxo de decisão de acesso entre o PEP e o PDP

O código 3.1 apresenta um exemplo de política ABAC utilizada na prova de

conceito, ilustrando a definição de regras de autorização e negação a partir de atributos do usuário, do recurso e do contexto de acesso.

Código 3.1: Exemplo de política ABAC utilizada no PDP

```
{
  "policyId": "P001",
  "effect": "Allow",
  "conditions": {
    "role": "admin",
    "resource_sensitivity": "high",
    "device_trusted": true,
    "network_type": ["LAN", "VPN"]
  }
}
```

Em seguida, as requisições aprovadas são submetidas ao modelo de aprendizado de máquina, que realiza uma análise contextual do pedido com o objetivo de estimar o nível de risco associado à tentativa de acesso. A partir desse risco estimado, o PDP define a decisão final de acesso, classificando a requisição em um dos três estados principais:

- *Allow*: o acesso é autorizado, indicando que o risco estimado é baixo e que os atributos do usuário, do recurso e do contexto estão alinhados às políticas de segurança definidas.
- *Deny*: o acesso é negado quando o risco estimado é elevado ou quando há indícios significativos de comportamento suspeito, mesmo após a aplicação das políticas ABAC.
- *Challenge*: o acesso é condicionado à verificação adicional, aplicada em situações de risco intermediário, exigindo, por exemplo, autenticação multifator (MFA) ou validação contextual antes da liberação.

Código 3.2: Exemplo de requisição de acesso enviada ao PEP

```
{
  "user_id": "1234",
  "role": "admin",
  "privilege_level": "low",
  "department": "engineering",
  "resource": "financial_report",
  "resource_sensitivity": "high",
  "timestamp": "2025-09-30T12:00:00",
```

```
  "network_type": "VPN",
  "location": "HQ",
  "device_type": "laptop",
  "os_version": "Ubuntu 22.04",
  "device_trusted": true
}
```

3.2.1 Geração e Tratamento dos Dados

A etapa experimental utilizou um conjunto de dados sintético gerado de forma programática, com o objetivo de simular cenários realistas de requisição de acesso em ambientes corporativos. Foram gerados aproximadamente 15 000 registros, cada um representando uma tentativa de acesso caracterizada por atributos de usuário, recurso e contexto, incluindo função do usuário, nível de privilégio, departamento, tipo e confiança do dispositivo, localização, tipo de rede, horário de acesso e sensibilidade do recurso.

A geração dos dados foi orientada por políticas determinísticas inspiradas em regras de acesso comuns em ambientes organizacionais. A partir da combinação dos atributos de usuário, recurso e contexto, foi definido um mecanismo heurístico de cálculo de risco, responsável por estimar um valor contínuo no intervalo $[0, 1]$. Esse valor de risco não foi utilizado como atributo de entrada, sendo empregado exclusivamente como critério para a rotulagem dos registros nas classes *allow*, *challenge* e *deny*. Para esse fim, foram definidos limiares determinísticos, de modo que valores de risco inferiores a 0,35 foram associados à classe *allow*, valores iguais ou superiores a 0,35 e inferiores a 0,60 à classe *challenge*, e valores iguais ou superiores a 0,60 à classe *deny*.

A função de risco atribui pesos diferenciados a fatores considerados críticos ou suspeitos, como acesso a recursos de produção a partir de dispositivos pessoais, uso de dispositivos não confiáveis, acessos remotos e horários atípicos. Pequenas variações aleatórias foram introduzidas para representar incerteza e variabilidade nos cenários simulados, evitando a geração de fronteiras de decisão excessivamente rígidas.

A Tabela 3.1 apresenta a distribuição das decisões no conjunto de dados sintético, evidenciando a presença de cenários de baixo, médio e alto risco. Essa distribuição foi definida de forma a evitar extremos de desbalanceamento, garantindo variedade suficiente para os experimentos e para a análise do comportamento do estimador de risco.

Tabela 3.1: Distribuição das decisões no conjunto de dados sintético

Decisão	**Proporção (%)**
allow	49,34
challenge	29,98
deny	20,68

Com o objetivo de garantir a reprodutibilidade dos experimentos, o processo de geração do conjunto de dados sintético utiliza uma semente fixa nos geradores de números pseudoaleatórios. Essa abordagem assegura que múltiplas execuções do experimento produzam o mesmo conjunto de dados, permitindo a validação consistente dos resultados e a comparação entre diferentes configurações do modelo.

3.2.2 Modelo de Estimativa de Risco

O componente de aprendizado de máquina desenvolvido neste trabalho tem como objetivo estimar o risco associado a uma tentativa de acesso, atuando como mecanismo de apoio à decisão no contexto da arquitetura Zero Trust. Diferentemente de abordagens em que modelos de aprendizado de máquina realizam diretamente a decisão de autorização, o modelo proposto não substitui as políticas ABAC, mas complementa o processo de decisão ao fornecer uma avaliação probabilística do contexto da requisição.

O modelo adotado foi uma árvore de decisão (*Decision Tree*), escolhida principalmente por sua interpretabilidade e pela facilidade de inspeção dos atributos utilizados no processo de predição. Essa característica é particularmente relevante em cenários de segurança, nos quais a explicabilidade contribui para auditoria e análise das decisões.

O conjunto de dados foi dividido automaticamente em subconjuntos de treinamento (75%) e teste (25%), por meio das funções da biblioteca *scikit-learn*, e empregado na construção de um modelo de árvore de decisão (*Decision Tree*), utilizado como estimador de risco contextual.

O treinamento do modelo foi realizado a partir do conjunto de dados sintético descrito na subseção anterior, no qual cada requisição contém atributos relacionados ao usuário, ao recurso acessado e ao contexto da solicitação. O modelo foi treinado como

um classificador multiclasse, com as classes *allow*, *challenge* e *deny*, de modo a aprender padrões presentes nos dados e permitir a extração de probabilidades associadas a cada uma dessas classes durante a fase de execução.

Em uma abordagem inicial, o risco foi calculado conforme a Equação 3.1, a partir das probabilidades associadas às classes de negação (*deny*) e de verificação adicional (*challenge*), considerando a saída probabilística do modelo de aprendizado de máquina. Essa formulação baseou-se em uma interpretação semântica das classes, na qual a decisão *deny* representaria cenários de risco elevado, enquanto *challenge* indicaria situações ambíguas que demandariam verificações adicionais, ao passo que *allow* seria associado a contextos considerados seguros.

$$R = P(deny) + 0.5 \cdot P(challenge) \tag{3.1}$$

Essa formulação mostrou-se inicialmente intuitiva e interpretável, pois atribui maior peso à probabilidade de negação de acesso, entendida como um indicativo de que o risco associado à requisição excede os limites aceitáveis. Tal interpretação está alinhada às abordagens tradicionais de controle de acesso e aos princípios da arquitetura Zero Trust descritos pelo NIST, nos quais decisões de negação ocorrem quando o contexto avaliado é considerado inseguro ou não confiável (ROSE et al., 2020).

Durante os experimentos, observou-se que a métrica era capaz de diferenciar cenários de forma relativa, atribuindo valores de risco mais elevados a contextos considerados menos seguros quando comparados a acessos normais.

Entretanto, apesar dessa ordenação coerente entre os cenários, os scores de risco permaneceram sistematicamente concentrados em valores muito baixos, tipicamente restritos a uma faixa próxima de zero. Tal comportamento está diretamente relacionado ao desbalanceamento natural do conjunto de dados sintéticos, no qual a classe *deny* ocorre com menor frequência em comparação às demais. Como consequência, o modelo tende a atribuir valores reduzidos a P(*deny*), limitando o impacto dessa classe no cálculo do risco e impedindo que a métrica explorasse de forma adequada todo o intervalo contínuo entre 0 e 1.

Para mitigar esse efeito, adotou-se uma reformulação conceitual do cálculo de

risco, passando a defini-lo como o complemento da probabilidade de autorização, isto é, risco $= 1 - P(\text{allow})$, conforme a Equação 3.2. Nessa abordagem, o risco deixa de representar apenas a probabilidade explícita de bloqueio e passa a refletir a falta de confiança do modelo em autorizar o acesso.

$$R = 1 - P(allow) \tag{3.2}$$

Do ponto de vista de implementação, o modelo é utilizado de forma distinta de um classificador tradicional. Durante a fase de execução, os atributos da requisição de acesso são previamente tratados e codificados conforme o mesmo esquema utilizado no treinamento, e então submetidos ao modelo por meio da função *predict_proba*. Essa função retorna uma distribuição de probabilidades associadas às classes aprendidas pelo modelo, refletindo o grau de confiança do classificador em cada possível resultado.

A probabilidade associada à classe *allow* é utilizada como insumo para o cálculo do risco, resultando em um score contínuo no intervalo $[0, 1]$. Dessa forma, valores mais elevados indicam menor confiança do modelo na autorização e, consequentemente, maior risco contextual associado à requisição. Essa interpretação mostra-se coerente com os princípios da arquitetura Zero Trust, nos quais a ausência de confiança explícita deve ser tratada como um sinal de risco, permitindo que incertezas e ambiguidades contextuais influenciem decisões adaptativas, como a exigência de autenticação adicional, mesmo quando não há indícios suficientes para um bloqueio direto.

A Figura 3.2 apresenta a distribuição dos scores de risco estimados pelo modelo para o conjunto de dados sintético. Observa-se uma concentração significativa de valores nas faixas inferiores de risco, indicando a predominância de cenários considerados legítimos ou de baixo risco no ambiente simulado. Esse comportamento é esperado em contextos corporativos, nos quais a maioria das requisições ocorre em condições regulares de uso.

Ao mesmo tempo, nota-se a presença de uma quantidade relevante de requisições com scores elevados, próximos ao limite superior do intervalo. Esses casos refletem cenários mais críticos, nos quais múltiplos fatores contextuais adversos se combinam, como acesso fora do horário comercial, uso de redes não confiáveis ou dispositivos não confiáveis. A distribuição resultante evidencia a capacidade do modelo em diferenciar contextos de

baixo, médio e alto risco, fornecendo um sinal contínuo adequado para orientar decisões adaptativas no PDP.

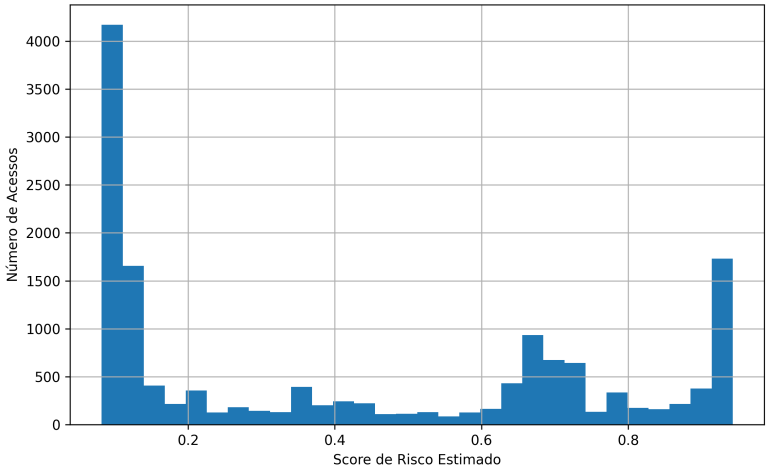

Figura 3.2: Distribuição dos scores de risco estimados pelo modelo

A Figura 3.3 apresenta a importância relativa dos atributos utilizados pelo modelo na estimativa do risco. Observa-se que características relacionadas ao contexto de acesso exercem maior influência do que atributos puramente identitários. O atributo *device_trusted* destaca-se como o fator mais relevante, indicando que a confiabilidade do dispositivo é um elemento central na avaliação de risco.

Outros atributos contextuais, como *time_of_day*, *network_type* e *resource_sensitivity*, também apresentam elevada importância, reforçando a influência de fatores temporais, de conectividade e da criticidade do recurso no cálculo do risco.

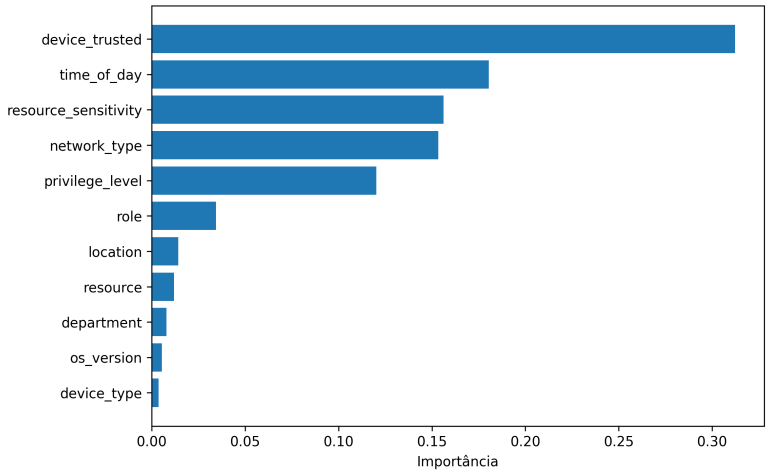

Figura 3.3: Importância relativa dos atributos na estimativa do risco

Com o objetivo de melhorar a confiabilidade das probabilidades estimadas, o classificador foi encapsulado utilizando a técnica de calibração de probabilidades por meio do método *sigmoid*, disponibilizado pela classe *CalibratedClassifierCV* da biblioteca *scikit-learn*. A calibração é especialmente relevante em conjuntos de dados sintéticos e de tamanho moderado, pois reduz vieses nas estimativas probabilísticas e torna o valor de risco mais consistente para uso em decisões de segurança.

A decisão final de acesso não é realizada pelo modelo de aprendizado de máquina. Após a estimativa do risco, o Policy Decision Point (PDP) aplica limiares predefinidos para classificar a requisição em *allow*, *challenge* ou *deny*. Especificamente, requisições com score de risco inferior a 0,35 são autorizadas (*allow*); valores de risco iguais ou superiores a 0,35 e inferiores a 0,60 resultam na exigência de verificação adicional (*challenge*); enquanto scores de risco iguais ou superiores a 0,60 levam à negação do acesso (*deny*).

No contexto desta PoC, a decisão *challenge* foi utilizada como um indicativo da necessidade de autenticação adicional, como autenticação multifator. Entretanto, não foi implementado um fluxo real de verificação MFA. Para fins de simplificação e foco na análise do mecanismo de decisão, considerou-se que requisições classificadas como *challenge* representariam acessos condicionados à validação bem-sucedida de um segundo fator de autenticação. Essa abordagem permitiu avaliar o comportamento da arquitetura e do modelo de estimativa de risco sem a complexidade adicional de integração com serviços externos de autenticação.

Dessa forma, o modelo de aprendizado de máquina atua como provedor de sinal de risco contínuo, enquanto a autoridade decisória permanece concentrada nas políticas de segurança e no fluxo de decisão da arquitetura Zero Trust, garantindo previsibilidade e controle sobre os critérios de autorização.

Essa abordagem combina a previsibilidade e o controle de políticas determinísticas com a capacidade adaptativa do aprendizado de máquina, resultando em um mecanismo de decisão mais flexível e alinhado aos princípios do Zero Trust.

3.2.3 Avaliação e Ajuste do Modelo

Foi desenvolvido um script para avaliar diferentes profundidades da árvore de decisão, com o objetivo de identificar um equilíbrio adequado entre desempenho e capacidade de generalização. A profundidade da árvore controla diretamente a complexidade do modelo: árvores com profundidade reduzida tendem a produzir regras excessivamente genéricas, podendo resultar em *underfitting*, enquanto árvores muito profundas tendem a memorizar padrões específicos do conjunto de treinamento, caracterizando o fenômeno de *overfitting*, no qual o modelo se ajusta excessivamente aos dados de treinamento, capturando ruídos ou particularidades do conjunto de treino e apresentando desempenho inferior quando aplicado a dados não vistos (GOODFELLOW; BENGIO; COURVILLE, 2016).

À medida que a profundidade aumenta, o modelo passa a criar divisões cada vez mais específicas no espaço de atributos, capturando combinações raras e altamente detalhadas. Embora isso possa elevar a acurácia no conjunto de treino, tais padrões nem sempre representam comportamentos gerais do domínio, o que compromete a capacidade de generalização do modelo.

A Figura 3.4 apresenta a variação da acurácia em função da profundidade da árvore, permitindo visualizar esse compromisso entre ajuste e generalização. Com base nessa análise, definiu-se a profundidade 8 como um ponto adequado, por apresentar desempenho satisfatório no conjunto de teste sem evidências de ajuste excessivo aos dados de treinamento.

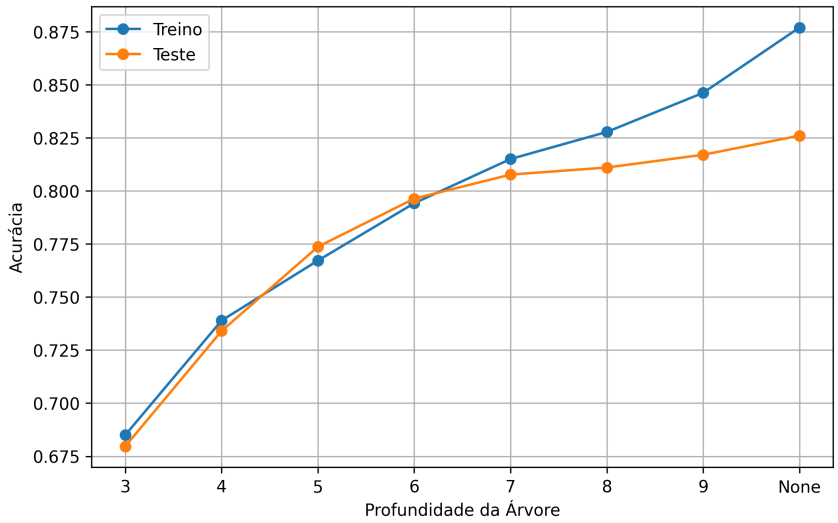

Figura 3.4: Variação da acurácia conforme a profundidade da árvore.

A Figura 3.5 ilustra a estrutura da árvore de decisão resultante, utilizada neste trabalho como estimador de risco contextual. A visualização da árvore permite observar como os atributos de entrada são combinados ao longo dos nós de decisão, evidenciando a forma como o modelo segmenta o espaço de atributos para estimar o risco associado às requisições de acesso no contexto da arquitetura Zero Trust.

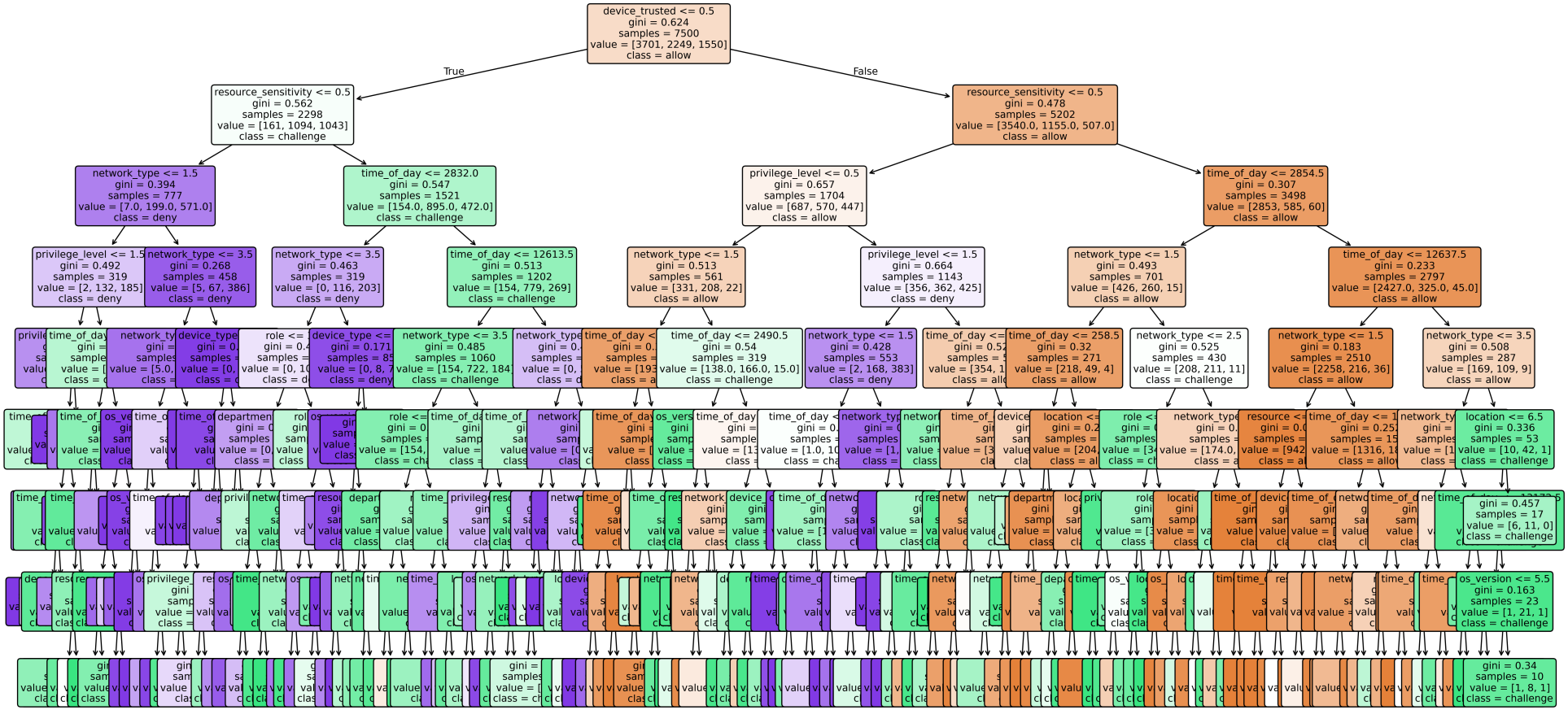

Figura 3.5: Visualização da árvore de decisão utilizada como estimador de risco contextual no processo de decisão Zero Trust.

A avaliação do modelo foi realizada por meio de métricas como acurácia, precisão, recall, f1-score e matriz de confusão, com o objetivo de analisar a qualidade das previsões probabilísticas produzidas pelo classificador e sua capacidade de distinguir padrões associados aos diferentes níveis de risco presentes no conjunto de dados. Os resultados obtidos são apresentados na Tabela 3.2.

Tabela 3.2: Relatório de Classificação do Modelo

Classe	**Precisão**	**Revocação**	**F1-Score**	**Suporte**
allow	0.86	0.92	0.89	1850
challenge	0.70	0.70	0.70	1124
deny	0.88	0.74	0.80	776
Acurácia			0.82	3750
Média Macro	0.81	0.79	0.80	3750
Média Ponderada	0.82	0.82	0.82	3750

3.2.4 Ferramentas e Tecnologias

A prova de conceito desenvolvida neste trabalho utilizou um conjunto de ferramentas e tecnologias voltadas à construção de aplicações orientadas a microsserviços, aprendizado de máquina e implantação em nuvem, com foco em reprodutibilidade e aderência ao modelo Zero Trust. As ferramentas adotadas podem ser organizadas de acordo com as duas etapas principais do desenvolvimento: a implementação e avaliação em ambiente local e a posterior adaptação da arquitetura para execução em nuvem.

Na etapa inicial, correspondente à implementação e avaliação em ambiente local, os componentes de Policy Enforcement Point e Policy Decision Point foram desenvolvidos em Python, utilizando o framework *FastAPI* para a construção das interfaces de comunicação baseadas em HTTP. O FastAPI foi escolhido por oferecer suporte nativo à validação de dados, alto desempenho e integração facilitada com modelos de aprendizado de máquina. Para a execução local da aplicação, foi utilizado o servidor ASGI *Uvicorn*, enquanto a biblioteca *httpx* foi empregada para a realização de chamadas HTTP entre os componentes da solução.

Ainda no contexto local, o tratamento e a análise dos dados sintéticos, bem como o

treinamento do modelo de estimativa de risco, foram realizados com as bibliotecas *pandas* e *numpy*. O modelo de aprendizado de máquina foi implementado com a biblioteca *scikit-learn*, por meio de uma árvore de decisão encapsulada com calibração de probabilidades. A persistência do modelo treinado e dos codificadores foi realizada utilizando a biblioteca *joblib*, enquanto a visualização dos resultados experimentais foi conduzida com a biblioteca *matplotlib*.

Na etapa seguinte, a solução foi adaptada para execução em um ambiente serverless na Amazon Web Services (AWS), com o objetivo de verificar a viabilidade da arquitetura proposta em um cenário distribuído. Essa adaptação permitiu a avaliação do comportamento da arquitetura fora do ambiente controlado local, mantendo a separação lógica entre os componentes e o fluxo de decisão definido.

A definição e o provisionamento da infraestrutura em nuvem foram realizados por meio da abordagem de *Infrastructure as Code* (IaC), utilizando a ferramenta Terraform. Essa estratégia possibilitou a automação da criação dos recursos necessários e contribuiu para a reprodutibilidade e o controle das configurações do ambiente experimental.

De forma geral, o conjunto de ferramentas adotado possibilitou a construção de uma arquitetura modular e desacoplada, alinhada aos princípios do Zero Trust, na qual políticas de segurança, estimativa de risco e aplicação de decisões são tratadas como componentes independentes, passíveis de evolução e substituição sem impacto direto nos demais módulos do sistema.

3.2.5 Adaptação da Arquitetura para Execução Serverless em Nuvem

A prova de conceito foi inicialmente desenvolvida e avaliada em ambiente local, utilizando um modelo tradicional de serviços executados de forma contínua. Com o objetivo de avaliar a arquitetura proposta em um cenário distribuído e aderente às práticas contemporâneas de computação em nuvem, a solução foi posteriormente adaptada para execução em um ambiente serverless na AWS.

Essa adaptação teve como princípio a preservação do modelo conceitual da arquitetura Zero Trust, mantendo a separação lógica entre os componentes de PEP e PDP,

bem como o fluxo de decisão baseado na aplicação sequencial de políticas determinísticas e na estimativa de risco contextual por meio de aprendizado de máquina. Assim, as alterações realizadas concentraram-se nos mecanismos de execução, inicialização e comunicação entre os componentes, sem impactar a lógica de decisão de acesso previamente implementada.

No ambiente local, a comunicação entre os componentes PEP e PDP era realizada por meio de chamadas HTTP síncronas. Na versão em nuvem, essa interação foi reformulada para utilizar mecanismos nativos da plataforma serverless, adotando a invocação direta de funções AWS Lambda. Essa abordagem eliminou a necessidade de comunicação HTTP interna entre serviços, reduzindo a dependência de camadas intermediárias e alinhando a arquitetura ao modelo orientado a eventos característico da AWS Lambda.

Adicionalmente, a forma de inicialização dos componentes precisou ser adaptada ao modelo de execução sob demanda típico de ambientes serverless, no qual as funções não permanecem em execução contínua. Nesse contexto, os componentes são instanciados de forma reativa a cada requisição de acesso, preservando o comportamento funcional observado na execução local e mantendo inalterada a interface de comunicação exposta aos clientes.

No que se refere à integração com a infraestrutura de nuvem, bibliotecas específicas foram incorporadas para viabilizar a execução da arquitetura nesse novo contexto. O adaptador *Mangum* foi empregado exclusivamente no componente PEP, permitindo a execução da aplicação desenvolvida em FastAPI — baseada no padrão ASGI — em funções AWS Lambda expostas via API Gateway. Por sua vez, o componente PDP foi implementado como uma função Lambda computacional pura, sem dependência de interfaces HTTP, sendo invocado diretamente pelo PEP por meio da biblioteca *boto3*.

Essa estratégia permitiu manter a coerência conceitual da arquitetura Zero Trust, ao mesmo tempo em que explorou de forma adequada os recursos da plataforma serverless. Como resultado, a arquitetura pôde ser avaliada em um ambiente de nuvem real, fornecendo subsídios para a análise comparativa entre a execução local e a execução serverless, apresentada na seção de Resultados.

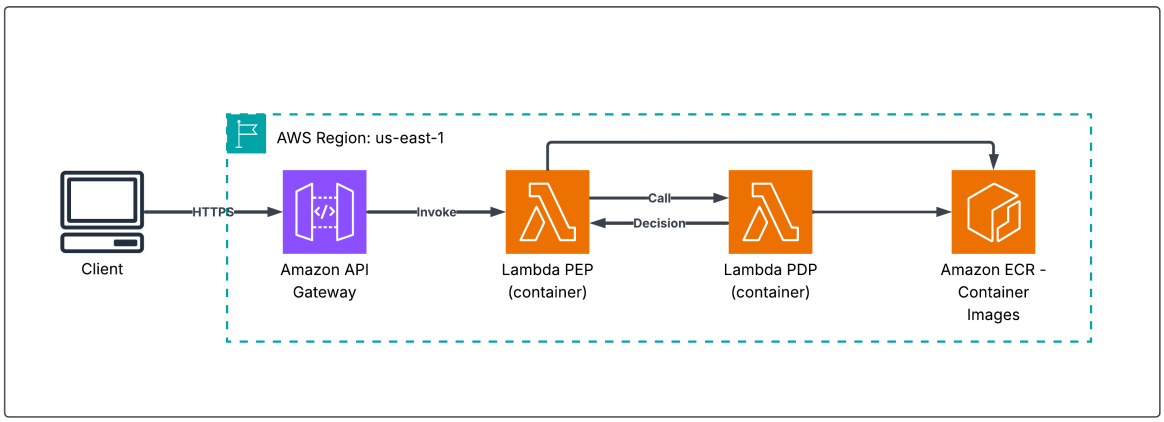

Figura 3.6: Arquitetura serverless do sistema Zero Trust na AWS com separação entre PEP e PDP

3.2.6 Disponibilidade do Código e Reprodutibilidade

A implementação completa da prova de conceito desenvolvida neste trabalho está disponível em um repositório público, permitindo a inspeção do código-fonte e a reprodução dos experimentos realizados. O código-fonte encontra-se disponível no GitHub[1].

Com o objetivo de facilitar a reprodutibilidade dos experimentos e a comparação entre os diferentes ambientes de execução, o repositório foi organizado em dois ramos principais (*branches*). O ramo *api_local* corresponde à versão executada em ambiente local, na qual a comunicação entre os componentes da arquitetura ocorre por meio de requisições HTTP tradicionais. Já o ramo *api_cloud* contém a versão adaptada para execução em ambiente serverless na AWS, utilizando funções AWS Lambda e mecanismos nativos de invocação direta entre os componentes Policy Enforcement Point e Policy Decision Point.

Essa organização permite isolar as adaptações específicas ao ambiente de nuvem, preservando a lógica de decisão de acesso e possibilitando a comparação direta dos resultados experimentais apresentados ao longo deste trabalho.

[1] ⟨https://github.com/1RodrigoSoares/ZeroTrustApi⟩

4 Resultados

4.1 Avaliação Experimental em Ambiente Local

Os resultados apresentados nesta subseção foram obtidos a partir da execução de testes funcionais e experimentais sobre a API responsável pela decisão de acesso. Os experimentos foram conduzidos em ambiente local, em condições controladas, com o objetivo de estabelecer uma linha de base para a avaliação do comportamento da arquitetura proposta, antes de sua adaptação para execução em ambiente serverless.

Cada cenário experimental consistiu no envio de requisições HTTP síncronas à interface de comunicação do serviço, contendo atributos do usuário, do recurso e do contexto, de modo a simular interações realistas entre clientes e o sistema de controle de acesso. As respostas retornadas pela API foram analisadas quanto à decisão final (*allow*, *challenge* ou *deny*), ao motivo associado à decisão e, quando aplicável, ao score de risco estimado pelo modelo de aprendizado de máquina. Essa abordagem permitiu avaliar o fluxo completo de decisão de acesso, incluindo a aplicação das políticas determinísticas baseadas em atributos, a estimativa de risco contextual e a lógica adaptativa implementada no PDP.

A execução local dos testes possibilitou a repetição controlada dos cenários, permitindo a verificação do determinismo e da reprodutibilidade das decisões ao longo de múltiplas execuções. Cenários idênticos produziram decisões consistentes e scores de risco equivalentes em execuções realizadas em momentos distintos, indicando estabilidade do modelo de aprendizado de máquina e previsibilidade do comportamento decisório em ambiente controlado.

No conjunto de cenários avaliados, observou-se que as políticas determinísticas baseadas em atributos atuaram como primeira camada de controle, bloqueando explicitamente requisições incompatíveis com regras de segurança previamente definidas. Aproximadamente um terço dos cenários resultou em decisões de negação direta (*deny*) nessa etapa, sem a necessidade de acionamento do modelo de aprendizado de máquina, caracte-

rizando o papel do ABAC como mecanismo de filtragem inicial de acessos potencialmente inseguros.

Os resultados também evidenciam a sensibilidade do modelo a variações contextuais. Alterações em atributos como horário de acesso, localização geográfica ou tipo de rede foram suficientes para impactar o score de risco estimado e, consequentemente, a decisão final. Em particular, acessos realizados fora do horário comercial, a partir de localizações incomuns ou por meio de redes menos confiáveis apresentaram maior propensão a decisões de *challenge*, mesmo quando o nível de privilégio do usuário e a sensibilidade do recurso não indicavam violação direta das políticas ABAC.

De forma geral, os resultados obtidos em ambiente local demonstram que a arquitetura proposta combina previsibilidade, assegurada pelas políticas determinísticas, com adaptabilidade contextual, viabilizada pela estimativa contínua de risco. Esses resultados servem como referência para a análise comparativa com a execução da arquitetura em ambiente serverless, apresentada na subseção seguinte.

Tabela 4.1: Resultados dos cenários de teste na execução local

Caso	**Tipo de Avaliação**	**Decisão Final**	**Tempo (ms)**
suspicious_remote_access	ML	deny	17.52
off_hours_high_privilege	ML	*allow*	14.91
mobile_network_access	ABAC	*deny*	1.39
vpn_weekend_access	ML	*allow*	14.97
normal_business_hours	ML	*challenge*	16.95
ml_failure_fallback	ABAC	*allow*	1.37
high_priv_low_sens_suspicious	ML	*deny*	17.52
trusted_device_unusual_context	ML	*challenge*	11.18
multiple_weak_signals	ABAC	*deny*	1.10
borderline_privilege_mismatch	ML	*allow*	14.92
holiday_emergency_access	ML	*allow*	11.46
repeated_request_v1	ML	*allow*	11.34
repeated_request_v2	ML	*challenge*	16.30
repeated_request_v3	ML	*challenge*	20.26
cross_department_access	ML	*challenge*	18.13
legacy_system_access	ML	*challenge*	18.00
perfect_storm_scenario	ABAC	deny	1.15

4.2 Avaliação Experimental em Ambiente Serverless na Nuvem

Após a adaptação da arquitetura para execução em ambiente serverless, foi conduzida uma avaliação experimental com o objetivo de analisar o comportamento do sistema em um cenário de nuvem real, considerando tanto a corretude das decisões de acesso quanto aspectos de desempenho associados à execução sob demanda. Os experimentos foram realizados por meio da API HTTP exposta pelo componente *Policy Enforcement Point*, implantada como função AWS Lambda integrada ao Amazon API Gateway.

Os mesmos cenários de teste definidos para a execução local foram reaplicados no ambiente serverless, preservando integralmente os dados de entrada e os critérios de avaliação. Ao todo, foram executados 17 casos de teste distintos, abrangendo decisões de permissão, negação e desafio, em contextos variados de usuário, recurso e ambiente. Todos os testes foram concluídos com sucesso, sem falhas de execução ou inconsistências funcionais.

No que se refere ao desempenho, o tempo médio de execução observado foi de aproximadamente 636 ms por requisição, com mediana de 633 ms. O menor tempo registrado foi de 551 ms, enquanto o maior atingiu cerca de 729 ms, resultando em um desvio padrão de 45,7 ms. Esses valores refletem o custo adicional inerente ao modelo serverless, que inclui a invocação da função, a serialização de dados e a execução do pipeline decisório completo, incluindo a avaliação baseada em aprendizado de máquina.

Tabela 4.2: Estatísticas de desempenho da *API serverless*

Métrica	**Valor**
Número total de testes	17
Tempo médio de execução	636.61 ms
Mediana	632.96 ms
Tempo mínimo	550.58 ms
Tempo máximo	728.73 ms
Desvio padrão	45.74 ms
Tempo médio (ABAC)	604.49 ms
Tempo médio (ML)	643.50 ms

A distribuição das decisões retornadas pelo sistema mostrou-se equilibrada, com 35,3% das requisições resultando em *allow*, 35,3% em *challenge* e 29,4% em *deny*. Observou-se que apenas 17,6% das decisões foram resolvidas exclusivamente por políticas determinísticas baseadas em ABAC, enquanto 82,4% envolveram a etapa adaptativa de avaliação de risco por meio do modelo de aprendizado de máquina, evidenciando o papel central desse componente no processo decisório.

A análise dos scores de risco gerados pelo modelo revelou uma clara correlação entre o valor estimado e a decisão final de acesso. As decisões de negação apresentaram média de risco significativamente mais elevada (0,655), enquanto as permissões ocorreram predominantemente em contextos de baixo risco (média de 0,208). As decisões classificadas como *challenge* situaram-se em uma faixa intermediária, com média de 0,450, corroborando o uso do risco como critério adaptativo para aplicação de controles adicionais.

De modo geral, os resultados indicam que a adaptação da arquitetura para o ambiente serverless preservou o comportamento funcional observado na execução local, ao mesmo tempo em que demonstrou viabilidade prática em um cenário de nuvem real. Embora o modelo serverless introduza uma latência adicional em relação à execução contínua local, os tempos observados permanecem compatíveis com aplicações de controle de acesso em tempo quase real, reforçando a adequação da abordagem proposta.

Tabela 4.3: Resultados dos cenários de teste na execução *serverless*

Caso	**Tipo de Avaliação**	**Decisão Final**	**Tempo (ms)**
suspicious_remote_access	ML	deny	550.58
off_hours_high_privilege	ML	*allow*	615.20
mobile_network_access	ABAC	*deny*	563.20
vpn_weekend_access	ML	*allow*	612.57
normal_business_hours	ML	*challenge*	597.95
ml_failure_fallback	ML	*allow*	669.24
high_priv_low_sens_suspicious	ML	*deny*	632.96
trusted_device_unusual_context	ML	*challenge*	640.78
multiple_weak_signals	ABAC	*deny*	629.33
borderline_privilege_mismatch	ML	*allow*	650.77
holiday_emergency_access	ML	*allow*	711.58
repeated_request_v1	ML	*allow*	617.44
repeated_request_v2	ML	*challenge*	728.73
repeated_request_v3	ML	*challenge*	668.60
cross_department_access	ML	*challenge*	668.29
legacy_system_access	ML	*challenge*	644.26
perfect_storm_scenario	ABAC	*deny*	620.94

4.2.1 Análise Comparativa entre Execução Local e Execução *Serverless*

A comparação entre os resultados obtidos em ambiente local e aqueles observados na execução *serverless* em nuvem evidencia diferenças relevantes, especialmente no que se refere ao tempo de execução das decisões de acesso. Enquanto a execução local apresentou tempos médios significativamente reduzidos, tipicamente na ordem de poucos milissegundos, a execução em ambiente *serverless* apresentou tempos médios em torno de centenas de milissegundos por requisição.

Essa diferença era esperada e decorre principalmente das características inerentes ao modelo de computação *serverless* adotado pela AWS. Mesmo com a utilização de mecanismos nativos da plataforma para otimizar a comunicação entre os componentes — como a invocação direta de funções AWS Lambda entre o PEP e o PDP, evitando

chamadas HTTP externas — a execução em nuvem envolve custos adicionais associados à serialização dos eventos, à inicialização do ambiente de execução e à orquestração interna da plataforma.

No ambiente local, tanto o PEP quanto o PDP são executados de forma contínua no mesmo contexto computacional, o que elimina sobrecargas relacionadas a cold starts, gerenciamento de infraestrutura e isolamento de execução. Já no ambiente *serverless*, cada requisição é tratada como um evento independente, exigindo a ativação do ambiente de execução e o carregamento dos recursos necessários para o processamento da decisão de acesso.

Apesar do aumento no tempo de resposta observado na execução em nuvem, os resultados demonstram que a arquitetura proposta mantém comportamento funcional consistente entre os dois ambientes. As decisões finais, a distribuição entre *allow*, *challenge* e *deny*, bem como a relação entre scores de risco e decisões adaptativas, permaneceram alinhadas com aquelas observadas no ambiente local.

Dessa forma, a análise comparativa indica que a adaptação da arquitetura para o modelo *serverless* introduz um custo adicional de latência, inerente ao modelo de execução sob demanda, sem comprometer a lógica decisória nem os princípios da Arquitetura Zero Trust. Tal *trade-off* entre desempenho e escalabilidade é característico de arquiteturas modernas baseadas em computação em nuvem, sendo aceitável no contexto de sistemas de controle de acesso que priorizam segurança, elasticidade e desacoplamento entre componentes.

5 Conclusões e trabalhos futuros

Os resultados obtidos demonstram que a combinação entre políticas ABAC e um modelo baseado em aprendizado de máquina é viável e funcional no contexto proposto. A abordagem baseada em políticas mostrou-se eficiente para decisões determinísticas e de baixo custo computacional, enquanto o modelo de aprendizado de máquina atuou como um mecanismo de apoio à decisão, capaz de capturar padrões contextuais mais sutis e fornecer uma estimativa contínua de risco associada às requisições de acesso. Essa combinação está alinhada aos princípios do Zero Trust, nos quais nenhuma requisição é implicitamente confiável e todas as decisões devem considerar o contexto de forma dinâmica.

Os experimentos indicaram que a formulação inicial do cálculo de risco, baseada na probabilidade da classe *deny*, resultava em scores sistematicamente baixos, em função do desbalanceamento do conjunto de dados sintéticos e da natureza supervisionada do modelo. Ainda assim, os cenários mais suspeitos apresentaram consistentemente os maiores valores relativos de risco. A reformulação do cálculo, ao considerar a incerteza do modelo por meio da probabilidade da classe *allow*, produziu uma distribuição mais adequada dos scores no intervalo $[0, 1]$, reforçando o papel do aprendizado de máquina como um mecanismo complementar de estimativa de risco às políticas ABAC em arquiteturas Zero Trust.

Os resultados devem ser interpretados à luz das limitações do estudo, em especial o uso de dados sintéticos, a adoção de um modelo de classificação relativamente simples e o tratamento conceitual da autenticação multifator, representada pela decisão *challenge* sem a implementação de um fluxo real de validação. Ainda assim, a prova de conceito desenvolvida contribui ao demonstrar, de forma prática e reprodutível, os desafios, decisões de projeto e implicações técnicas associadas à incorporação de aprendizado de máquina em mecanismos de decisão de acesso baseados em Zero Trust.

Como trabalhos futuros, destaca-se a utilização de conjuntos de dados reais de acesso, provenientes de ambientes corporativos ou sistemas em produção, de modo a avaliar o comportamento do modelo em cenários mais próximos da realidade operacional.

Além disso, a investigação de modelos de aprendizado de máquina mais robustos e de técnicas de calibração de probabilidades pode contribuir para a melhoria da qualidade das estimativas de risco. Por fim, a integração com provedores de identidade e serviços de MFA, bem como avaliações de desempenho e escalabilidade em ambientes *serverless*, representam extensões naturais deste trabalho.

Bibliografia

ALBOQMI, R.; JAHAN, S.; GAMBLE, R. F. A runtime trust evaluation mechanism in the service mesh architecture. In: *2023 10th International Conference on Future Internet of Things and Cloud (FiCloud)*. [S.l.: s.n.], 2023. p. 242–249.

EXAME. *Ataques cibernéticos geram perdas de US$ 12 bi ao setor financeiro em duas décadas, diz FMI*. 2024. Acesso em: 27 jul. 2025. Disponível em: ⟨https://exame.com/economia/ataques-ciberneticos-geram-perdas-de-us-12-bi-ao-setor-financeiro-em-duas-decadas-diz-fmi/⟩.

FALCãO, E. et al. Supporting confidential workloads in spire. In: *2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. [S.l.: s.n.], 2022. p. 186–193.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. ⟨http://www.deeplearningbook.org⟩.

KANG, H. et al. Theory and application of zero trust security: A brief survey. *Entropy*, v. 25, p. 1595, 11 2023.

KHOLIDY, H. A. et al. Secure the 5g and beyond networks with zero trust and access control systems for cloud native architectures. In: *2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*. [S.l.: s.n.], 2023. p. 1–8.

KINDERVAG, J. *No More Chewy Centers: Introducing the Zero Trust Model of Information Security*. Cambridge, MA, USA, 2010.

MILLER, L. et al. Towards secure and leak-free workflows using microservice isolation. In: *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*. [S.l.: s.n.], 2021. p. 1–5.

MONTEBUGNOLI, S.; SABBIONI, A.; FOSCHINI, L. Evaluating mesh communications in disaggregated near-rt ric for 5g open ran: a functional and performance analysis. In: *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*. [S.l.: s.n.], 2024. p. 571–576.

NORTHCUTT, S. et al. *Inside Network Perimeter Security (Inside)*. 2nd. ed. Indianapolis, IN, USA: Sams, 2005.

POUDEL, A. et al. Mazu: A zero trust architecture for service mesh control planes. In: *Proceedings of the 18th European Workshop on Systems Security*. New York, NY, USA: Association for Computing Machinery, 2025. (EuroSec'25), p. 49–55. ISBN 9798400715631. Disponível em: ⟨https://doi.org/10.1145/3722041.3723100⟩.

PRISMA Group. *PRISMA 2020 Statement*. 2020. Acesso em: 12 jun. 2025. Disponível em: ⟨http://www.prisma-statement.org/PRISMAStatement/FlowDiagram⟩.

ROSE, S. et al. *Zero Trust Architecture*. [S.l.], 2020.

VISWANATHAN, J.; N, D. K.; KUMAR, S. U. Zero trust security for web applications in microservice-based environments. In: *2024 First International Conference on Data, Computation and Communication (ICDCC)*. [S.l.: s.n.], 2024. p. 488–494.

WOHLIN, C. et al. *Experimentation in Software Engineering*. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434.