MBA EM **ENGENHARIA DE SOFTWARE**

Autenticação e Segurança em Microsserviços

Prof. José Maria Cesário Jr.

MBA USP ESALQ

A responsabilidade pela idoneidade, originalidade
e licitude dos conteúdos didáticos apresentados é do professor.

MBA USP ESALQ

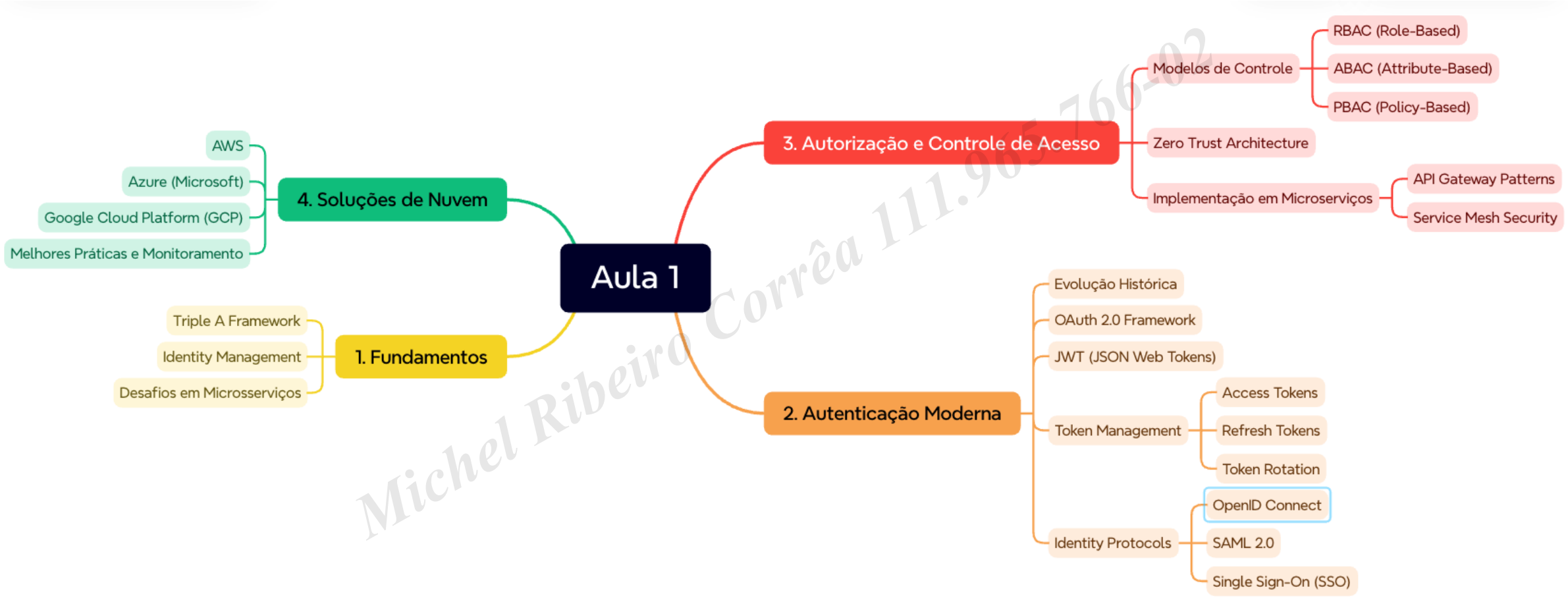
José Maria Cesário Júnior

- José Maria Cesário Júnior possui mais de 25 anos de experiência na área de TI e atua como Arquiteto de Soluções para Parceiros na AWS Brasil.
- Possui experiência em soluções de Cloud Computing, Indústria 4.0 e IoT Industrial
- Atua desde 2007 como professor universitário de Graduação e Pós Graduação.
- Analista de Sistemas, graduado pela Universidade Metodista de Piracicaba, com especialização em Tecnologia
- MBA em Gestão de Projetos pela FGV
- Especialista em Automação e Controle de Processos Industriais e Agroindustriais pela FEAGRI, UNICAMP
- Mestre em Tecnologia e Inovação pela UNICAMP

https://www.linkedin.com/in/josemariacesariojunior/

AGENDA MICROSSERVIÇOS

Data	Conteúdo
14/08/2025	Fundamentos Autenticação Moderna Autorização e Controle de Acesso Soluções de Nuvem Exercícios

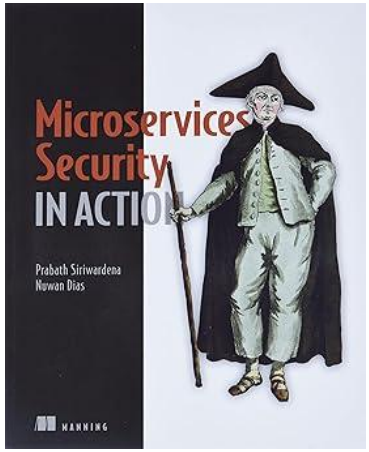


ÍCONES

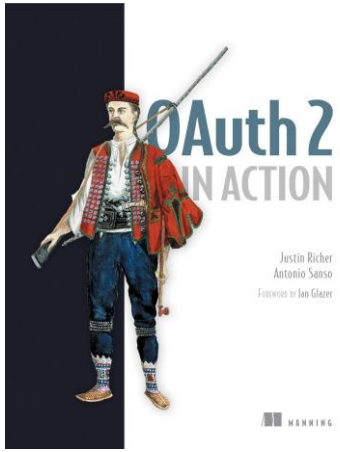
DEFINIÇÃO OU PONTO DE ATENÇÃO

EXERCÍCIO OU ATIVIDADE PRÁTICA

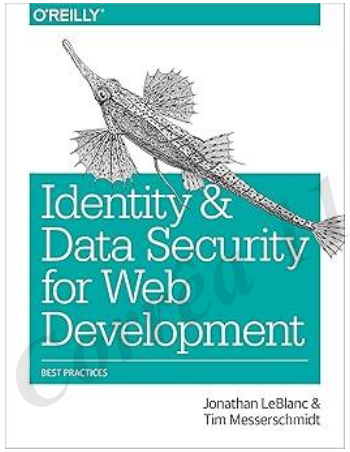
LIVROS



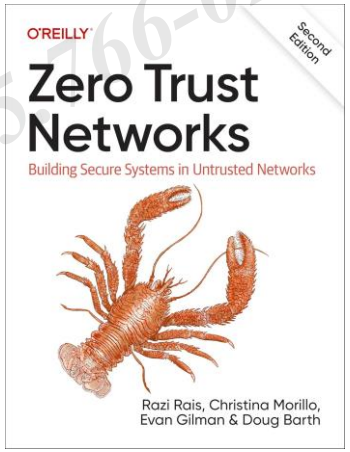
Microservices Security in Action
Agosto 2020
Edição Inglês
Nuwan Dias
Prabath Siriwardena



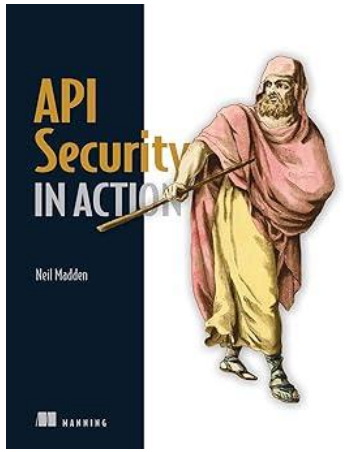
OAuth 2 in Action
2017
Edição Inglês
Justin Richer
Antonio Sanso



Identity and Data Security for Web Development
Julho 2016
Edição Inglês
Jonathan LeBlanc
Tim Messerschmidt



Zero Trust Networks
Abril 2024
Edição Inglês
Razi Rais
Christina Morillo
Evan Gilman
Doug Barth



API Security in Action
English Edition
eBook
Neil Madden

ACESSO AO AMBIENTE NA NUVEM

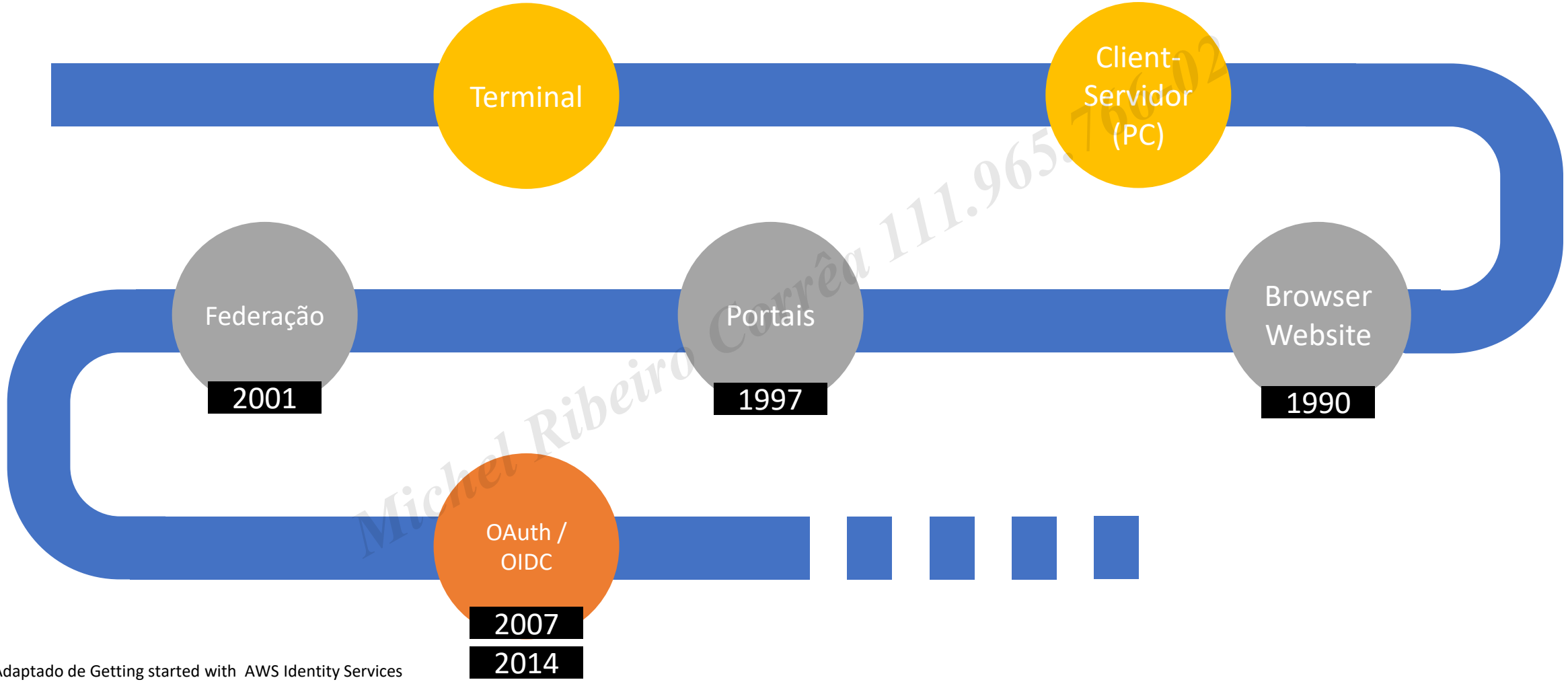
⚠️ IMPORTANTE ⚠️

- O laboratório prático utilizará recursos da AWS (Amazon Web Services) que podem gerar custos reais. Pontos essenciais:
- A participação nesta atividade é OPCIONAL
- Cada aluno é responsável pelo uso de sua própria conta AWS
- A AWS oferece um nível gratuito (Free Tier) com limites específicos
- Custos além do Free Tier serão de responsabilidade individual do usuário
- Recomendamos verificar os limites do Free Tier antes de iniciar as atividades práticas em https://aws.amazon.com/pt/free/

Fundamentos de Segurança

Parte 1

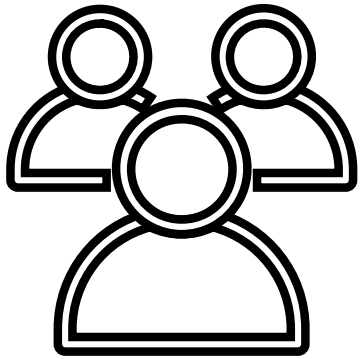
Uma história resumida da identidade



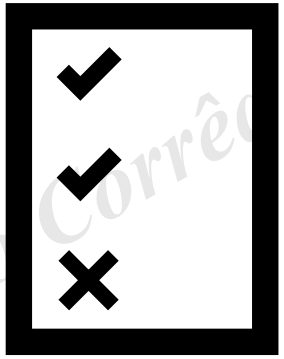
Fonte: Adaptado de Getting started with AWS Identity Services

Identidade, acesso e gerenciamento de recursos

Quem **pode acessar** **o que**



Gerenciamento de Identidade

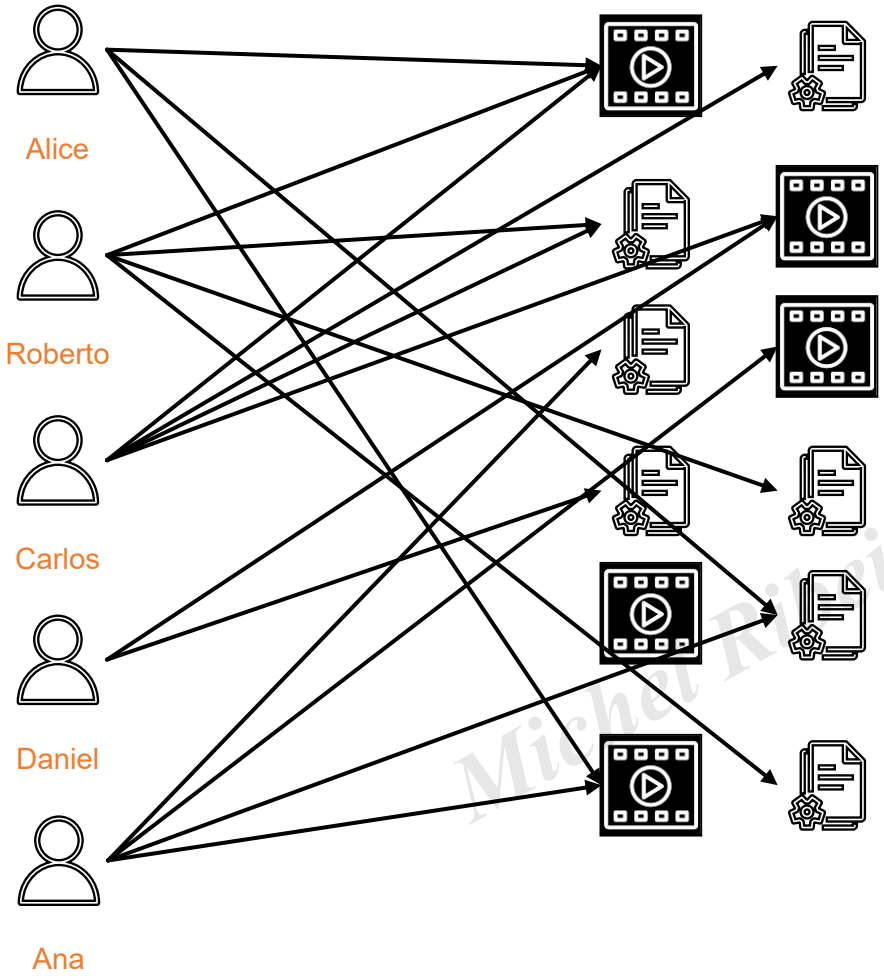


Gerenciamento de Acesso

Gerenciamento de Recursos

Fonte: Adaptado de Getting started with AWS Identity Services

Problemas iniciais

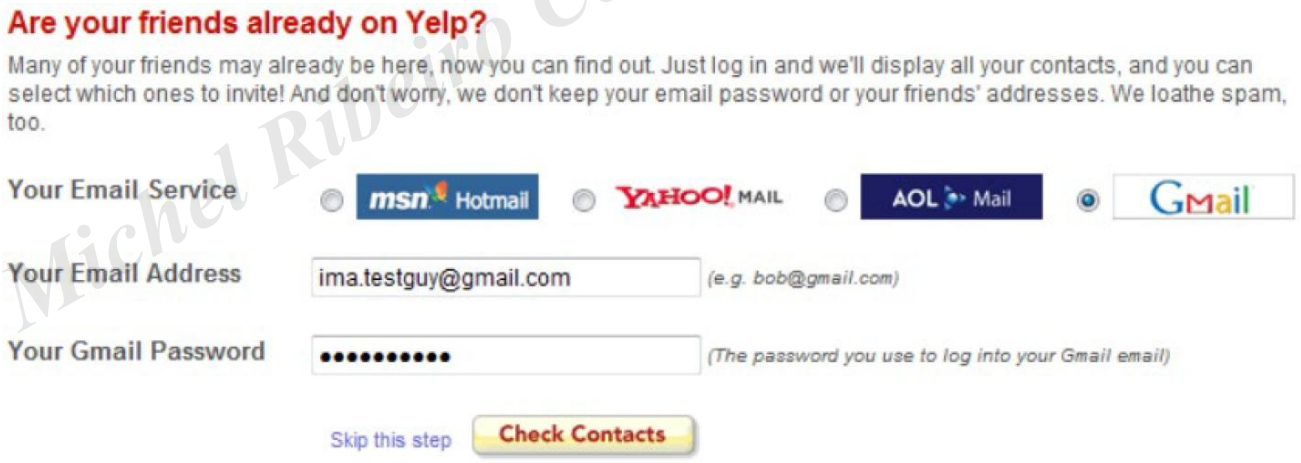


Fonte: Adaptado de Getting started with AWS Identity Services

- Múltiplas contas
- Má experiência do Usuário
- Falta de controle centralizado pelo usuário
- Promoção da reutilização de senhas
- Sem interoperabilidade padronizada
- Segurança proprietária não comprovada
- Segurança através da obscuridade
- Integração complexa de engenharia
- Falta de ferramentas comuns

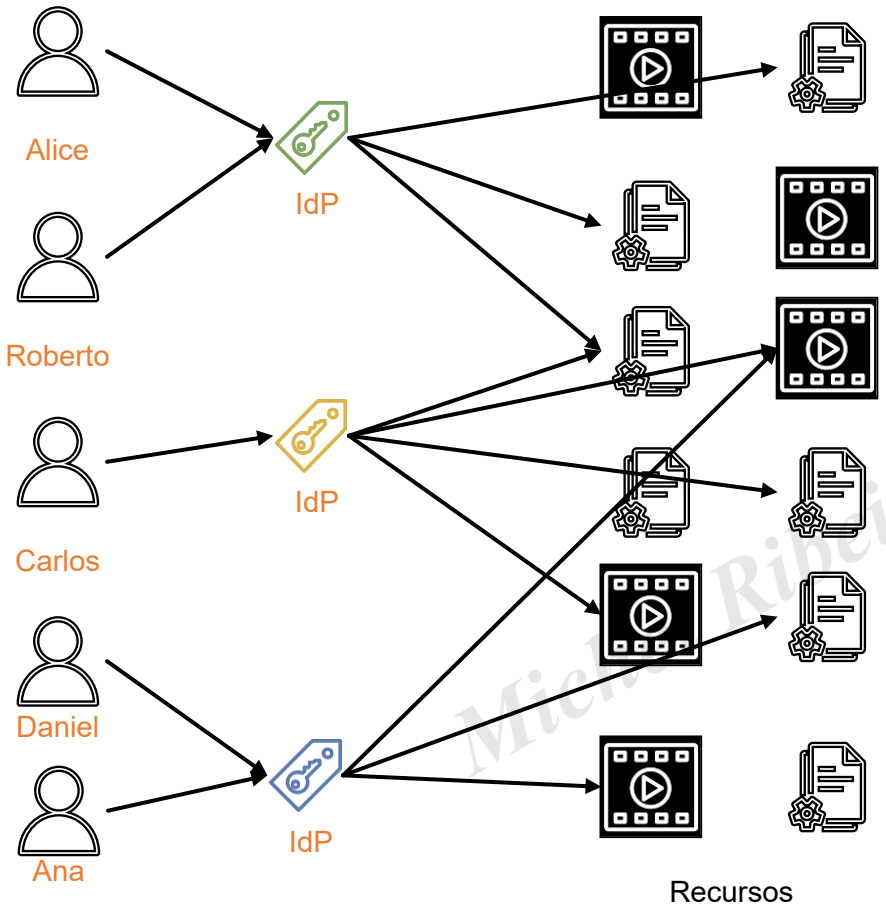
Concedendo acesso antes dos padrões modernos

- No início dos anos 2000, muitos novos serviços estavam sendo criados, expondo APIs que permitiam o compartilhamento de dados entre organizações – por exemplo, catálogos de endereços, dados contábeis, aplicativos de calendário.



Fonte: Adaptado de Getting started with AWS Identity Services

Uma abordagem melhor – Autenticação e autorização centralizadas



- Experiência do usuário aprimorada
- Autenticação federada, conjunto único de credenciais
- Identidade/Conta Única
- Controle centralizado pelo usuário
- Interoperabilidade baseada em Padrões Abertos
- Segurança Examinada – funcionalidade documentada publicamente
- Disponibilidade de ferramentas, APIs e SDKs comuns
- Integração de engenharia mais fácil

Fonte: Adaptado de Getting started with AWS Identity Services

Por que Segurança em Microsserviços?

- A transição de arquiteturas monolíticas para microsserviços introduz complexidades significativas na implementação de segurança. Enquanto em um monólito temos um perímetro de segurança bem definido, em microsserviços precisamos lidar com múltiplos pontos de entrada, comunicação inter-serviços e distribuição de responsabilidades de segurança.
- Este cenário exige uma abordagem mais sofisticada, onde cada serviço deve ser capaz de validar identidades, autorizar acessos e manter logs de auditoria de forma independente, mas coordenada.
- A superfície de ataque em microsserviços é exponencialmente maior, pois cada serviço representa um potencial ponto de vulnerabilidade. Além disso, a natureza distribuída dessas arquiteturas torna mais desafiador manter consistência nas políticas de segurança e detectar ataques que podem se propagar através de múltiplos serviços.

Fonte: NIST SP 800-204 - Security Strategies for Microservices-based Application Systems

O Framework Triple A (AAA)

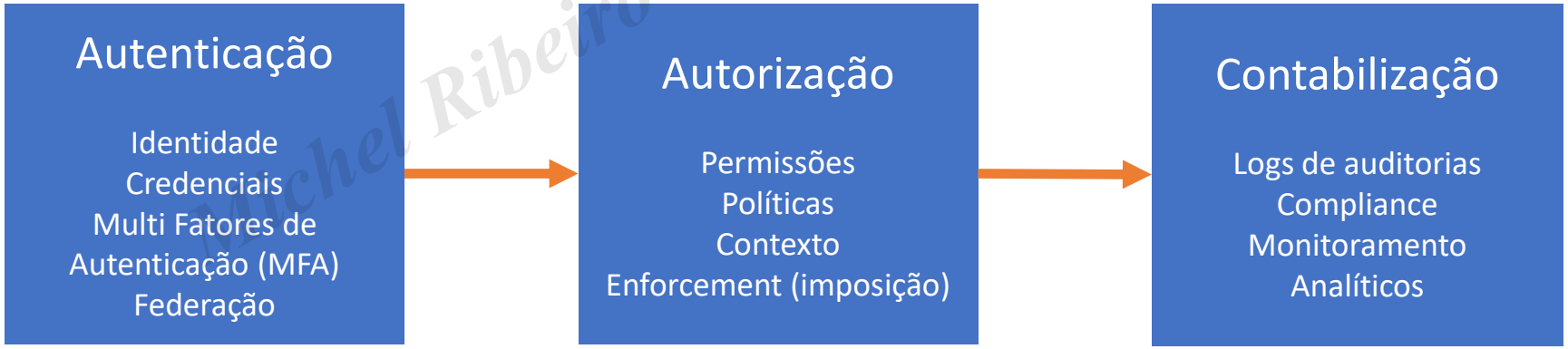
- O framework Triple A representa os três pilares fundamentais de qualquer sistema de segurança robusto

Authentication (Autenticação)	Authorization (Autorização)	Accounting (Contabilização ou Registro de atividades)
• Responde à pergunta "Quem é você?", sendo o processo de verificação da identidade de um usuário ou sistema através de credenciais como senhas, certificados digitais ou tokens biométricos.	• Determina "O que você pode fazer?", controlando quais recursos um usuário autenticado pode acessar e quais operações pode executar.	• Registra "O que você fez?", mantendo logs detalhados de todas as ações para auditoria, compliance e detecção de anomalias.

Fonte: NIST SP 800-204 - Security Strategies for Microservices-based Application Systems

O Framework Triple A (AAA)

- Em microsserviços, cada componente do Triple A deve ser implementado de forma distribuída, mantendo consistência e performance.
- A autenticação pode ser centralizada através de um Identity Provider, mas a autorização frequentemente precisa ser implementada em cada serviço individual, considerando o contexto específico de cada operação.



Fonte: NIST SP 800-204 - Security Strategies for Microservices-based Application Systems eRFC 4949 - Internet Security Glossary, Version 2

Authentication (Autenticação) - Verificação de Identidade

- A autenticação em microsserviços vai além da simples verificação de credenciais, envolvendo múltiplas camadas de validação que devem funcionar de forma eficiente em um ambiente distribuído.
- Os fatores de autenticação incluem algo que você sabe (knowledge factors como senhas), algo que você possui (possession factors como tokens ou smartphones) e algo que você é (inherence factors como biometria).
- A implementação de Multi-Factor Authentication (MFA) torna-se crítica em microsserviços devido à maior superfície de ataque.

Fonte: NIST SP 800-63B - Authentication and Lifecycle Management

Authentication (Autenticação) - Verificação de Identidade

- Em arquiteturas distribuídas, a autenticação deve considerar não apenas usuários finais, mas também a comunicação service-to-service.
- Isso inclui mutual TLS (mTLS) para autenticação bidirecional entre serviços, service accounts para identidades de aplicação e certificados digitais para validação de integridade. O desafio principal é manter a performance enquanto garante que cada requisição seja adequadamente autenticada sem criar gargalos no sistema.

Fonte: NIST SP 800-63B - Authentication and Lifecycle Management

Authorization (Autorização) - Controle de Acesso

- A autorização em microsserviços representa um dos maiores desafios arquiteturais, pois deve balancear granularidade de controle com performance e simplicidade de gestão.
- Diferentemente de sistemas monolíticos onde a autorização pode ser centralizada, em microsserviços cada serviço frequentemente precisa tomar decisões de autorização baseadas em seu contexto específico.
- Isso requer a implementação do princípio do menor privilégio, onde cada usuário ou serviço recebe apenas as permissões mínimas necessárias para executar suas funções.

Fonte: OWASP Application Security Verification Standard v4.0

Authorization (Autorização) - Controle de Acesso

- A separação de responsabilidades torna-se fundamental, onde diferentes serviços podem ter diferentes modelos de autorização baseados em suas necessidades específicas.
- Por exemplo, um serviço de pagamentos pode implementar controles mais rigorosos que um serviço de catálogo de produtos.
- A autorização deve considerar não apenas quem está fazendo a requisição, mas também o contexto da requisição, incluindo localização geográfica, horário, dispositivo utilizado e padrões de comportamento histórico.

Fonte: OWASP Application Security Verification Standard v4.0

Contabilização - Auditoria e Rastreamento

- O componente de accounting em microsserviços vai muito além do simples logging, representando um sistema complexo de rastreabilidade e observabilidade que deve funcionar de forma distribuída.
- Em um ambiente onde uma única transação de usuário pode atravessar dezenas de microsserviços, manter a correlação e rastreabilidade de eventos é fundamental para auditoria e detecção de anomalias. Isso inclui a implementação de correlation IDs que permitem rastrear uma requisição através de toda a arquitetura distribuída.

Fonte: ISO/IEC 27001:2013 - Information Security Management

Contabilização - Auditoria e Rastreamento

- A conformidade com regulamentações como LGPD, GDPR e SOX exige que todos os acessos a dados pessoais sejam registrados com detalhes suficientes para auditoria. Em microsserviços, isso significa que cada serviço deve contribuir para um log de auditoria unificado, mantendo informações sobre quem acessou quais dados, quando, de onde e que operações foram realizadas.
- A implementação de métricas de segurança em tempo real permite a detecção proativa de padrões suspeitos e resposta rápida a incidentes.

Fonte: ISO/IEC 27001:2013 - Information Security Management

Identity Management - Gestão de Identidades Distribuídas

- O gerenciamento de identidades em microsserviços requer uma abordagem sofisticada que vai além dos tradicionais diretórios centralizados. O lifecycle management de identidades deve considerar a criação, atualização, desativação e eventual remoção de identidades de forma que seja consistente através de todos os serviços da arquitetura.
- Isso inclui não apenas identidades humanas (usuários), mas também identidades de serviços, dispositivos IoT, e sistemas externos que interagem com a plataforma.
- A Identity Federation permite que organizações integrem múltiplos domínios de identidade, possibilitando que usuários de diferentes organizações acessem recursos de forma segura sem necessidade de múltiplas credenciais.

Fonte: NIST SP 800-63 - Digital Identity Guidelines

Identity Management - Gestão de Identidades Distribuídas

- Em microsserviços, isso é particularmente importante quando diferentes serviços podem ter diferentes provedores de identidade ou quando a arquitetura precisa integrar com sistemas legados.
- Directory Services como LDAP e Active Directory continuam relevantes, mas devem ser integrados através de protocolos modernos como SCIM (System for Cross-domain Identity Management).

Fonte: NIST SP 800-63 - Digital Identity Guidelines

Access Management - Controle de Acesso Contextual

- O gerenciamento de acesso em microsserviços deve ser dinâmico e contextual, considerando não apenas a identidade do usuário, mas também uma ampla gama de fatores contextuais que podem influenciar as decisões de acesso.
- Isso inclui localização geográfica, horário de acesso, dispositivo utilizado, padrões de comportamento histórico e até mesmo indicadores de risco em tempo real.
- O session management em ambientes distribuídos apresenta desafios únicos, pois sessões devem ser compartilhadas entre múltiplos serviços mantendo consistência e performance.
- Os padrões de acesso podem ser implementados como push (onde políticas são distribuídas para os pontos de enforcement) ou pull (onde cada ponto de enforcement consulta um serviço central de políticas).

Fonte: Gartner Magic Quadrant for Access Management

Access Management - Controle de Acesso Contextual

- A delegação de autorização permite que usuários concedam acesso limitado a terceiros sem compartilhar suas credenciais, sendo fundamental em arquiteturas que integram com parceiros externos. Context-aware access considera fatores como localização, dispositivo, horário e comportamento para tomar decisões de acesso mais inteligentes e seguras.

Fonte: Gartner Magic Quadrant for Access Management

Resource Management - Proteção de Recursos Distribuídos

- A proteção de recursos em microsserviços requer uma abordagem em camadas que considera tanto a segurança individual de cada serviço quanto a segurança da arquitetura como um todo.
- O API Gateway atua como o primeiro ponto de controle, implementando políticas de rate limiting, throttling e validação básica antes que as requisições alcancem os serviços internos.
- Cada microserviço deve implementar suas próprias validações específicas, criando múltiplas camadas de proteção que seguem o princípio de defense in depth.
- O isolamento de recursos entre tenants é fundamental em arquiteturas multi-tenant, onde diferentes organizações ou grupos de usuários compartilham a mesma infraestrutura mas devem ter acesso apenas aos seus próprios dados.

Fonte: OWASP API Security Top 10

Resource Management - Proteção de Recursos Distribuídos

- A classificação de dados permite implementar diferentes níveis de proteção baseados na sensibilidade das informações, desde dados públicos até informações altamente confidenciais que requerem criptografia adicional e controles de acesso mais rigorosos.
- A implementação de padrões como circuit breakers ou bulkhead ajudam a prevenir que falhas de segurança em um serviço se propaguem para outros componentes da arquitetura.

Fonte: OWASP API Security Top 10

Desafios Específicos em microsserviços

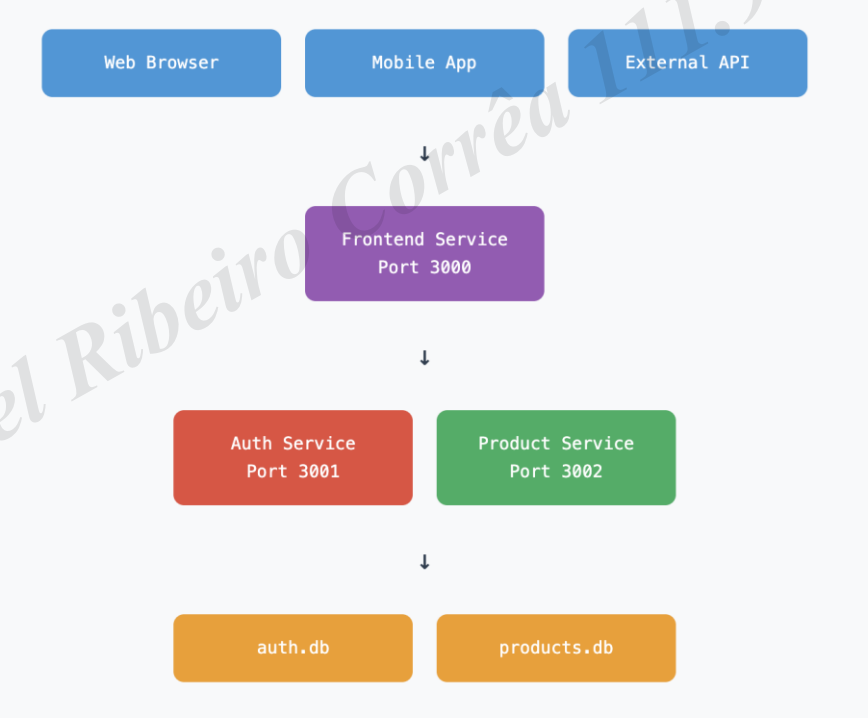
- A implementação de segurança em microsserviços apresenta desafios únicos que não existem em arquiteturas monolíticas.

Distribuição de estado de segurança	Latência de rede	Consistência de políticas de segurança	Observabilidade de segurança
• A distribuição de estado de segurança através de múltiplos serviços cria complexidades na manutenção de consistência, especialmente quando diferentes serviços podem ter diferentes modelos de dados e políticas de segurança	• A latência da rede torna-se um fator crítico, pois cada validação de segurança que requer comunicação entre serviços adiciona overhead que pode impactar significativamente a performance do sistema	• A consistência de políticas de segurança através de dezenas ou centenas de microsserviços requer ferramentas e processos sofisticados de governança • Mudanças em políticas devem ser propagadas de forma coordenada, e a verificação de compliance deve ser automatizada para garantir que todos os serviços estejam aderentes às políticas organizacionais	• A observabilidade de segurança torna-se exponencialmente mais complexa quando eventos de segurança podem estar distribuídos através de múltiplos serviços, logs e sistemas de monitoramento

Fonte: Microservices.io - Security Patterns

Exercício

- Entender e revisar a app de E-Commerce de acordo com os fundamentos de Autenticação e Segurança



Autenticação Moderna

Parte 2

Como aplicações modernas gerenciam autenticação ?

Autenticação Básica
(Basic Authentication)

Portador de Tokens
(Bearer Tokens)

Tokens JWT

Autenticação OAuth2

Tokens Access e Refresh

Single Sign-On (SSO) e
Protocolos de Identidade

Fonte: OWASP API Security Top 10

Token

PhotoMelon/iStock/Getty Images Plus/GettyImages

C1 UK

(US **gift certificate**)

a round metal or plastic disc that can be used instead of money :

- *Metal tokens are used instead of cash in some slot machines in casinos.*
- *First they had to queue up to receive a small plastic token, which they could then exchange for food.*

Um disco redondo de metal ou plástico que pode ser usado no lugar de dinheiro:

- Fichas de metal são usadas no lugar de dinheiro em algumas máquinas caça-níqueis de cassinos.

token *noun* [C] **(COMPUTING)**

COMPUTING • specialized

a piece of data that is used to represent and replace another one, in order to prevent private information being seen by someone who is not allowed to do so :

- *When we receive the transaction securely, we tokenize it and send back a token to the merchant so they never have to see a card number for settlement.*

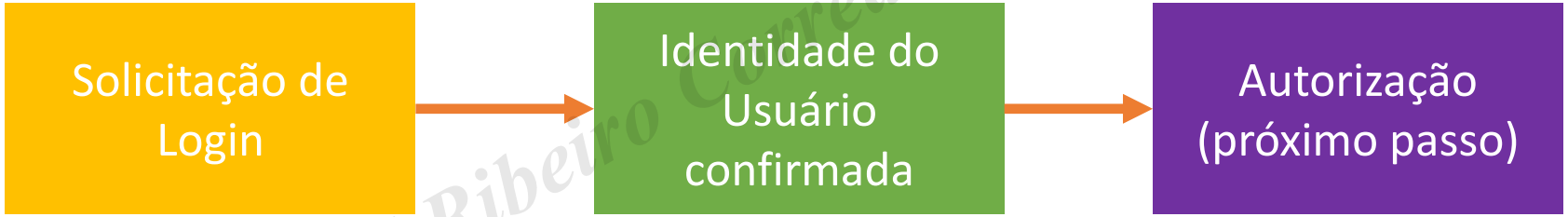
Um dado que é usado para representar e substituir outro, a fim de evitar que informações privadas sejam vistas por alguém que não tem permissão para fazê-lo:

- Quando recebemos a transação com segurança, nós a tokenizamos e enviamos um token de volta ao comerciante para que ele nunca precise ver o número do cartão para liquidação.

Fonte: https://dictionary.cambridge.org/pt/dicionario/ingles/token

O que autenticação ?

Atenticação = Quem você é
Primeiro passo antes da autorização

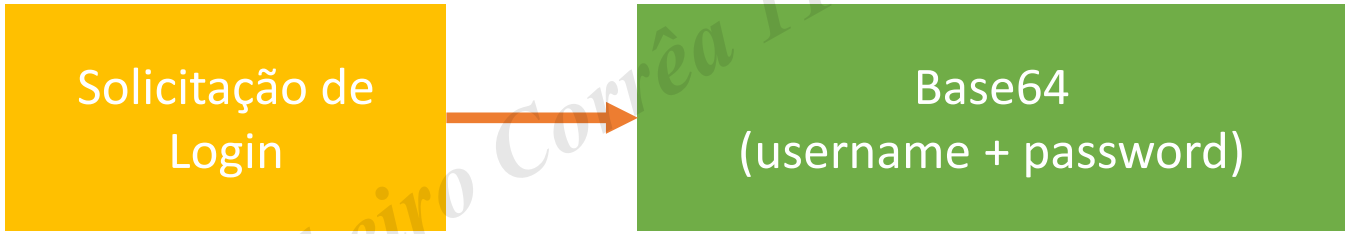


Antes que você posssa acessar dados ou realizar ações, o sistema precisa saber quem você é

Fonte: Adaptado de https://www.youtube.com/watch?v=9JPnN1Z_iSY

Autenticação Básica (Basic Authentication)

Nome de Usuário + Senha
Basic base64

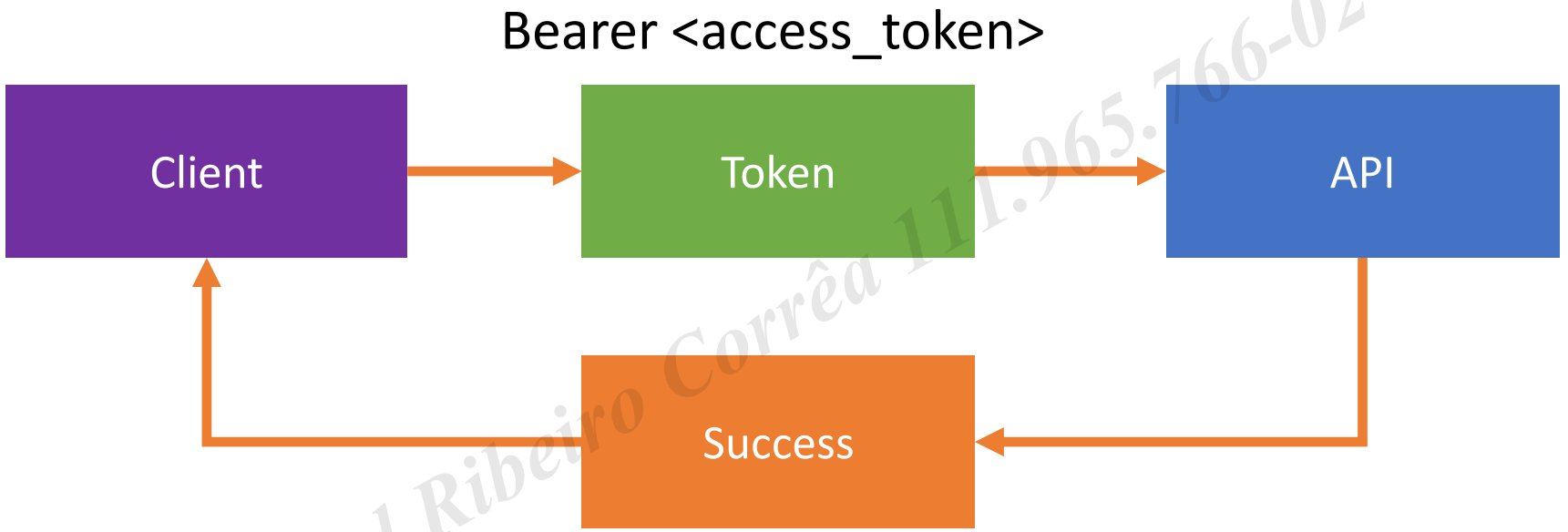


O token apenas é codificado em base64, uma representação textual de um conjunto de bytes

Raramente utilizado, solução simples e **insegura**, a menos que se use HTTPS

Fonte: Adaptado de https://www.youtube.com/watch?v=9JPnN1Z_iSY

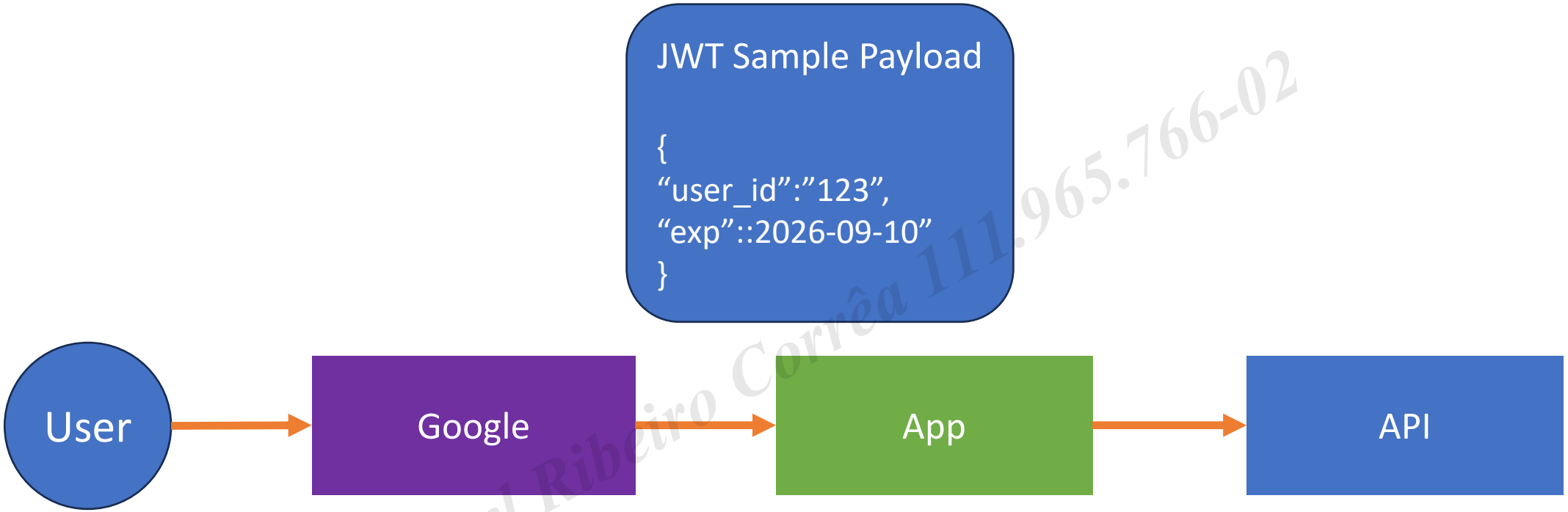
Autenticação Bearer Token



Abordagem padrão no projeto de API atualmente
Rápida e stateless (sem estado)

Fonte: Adaptado de https://www.youtube.com/watch?v=9JPnN1Z_iSY

OAuth2 e JWTs

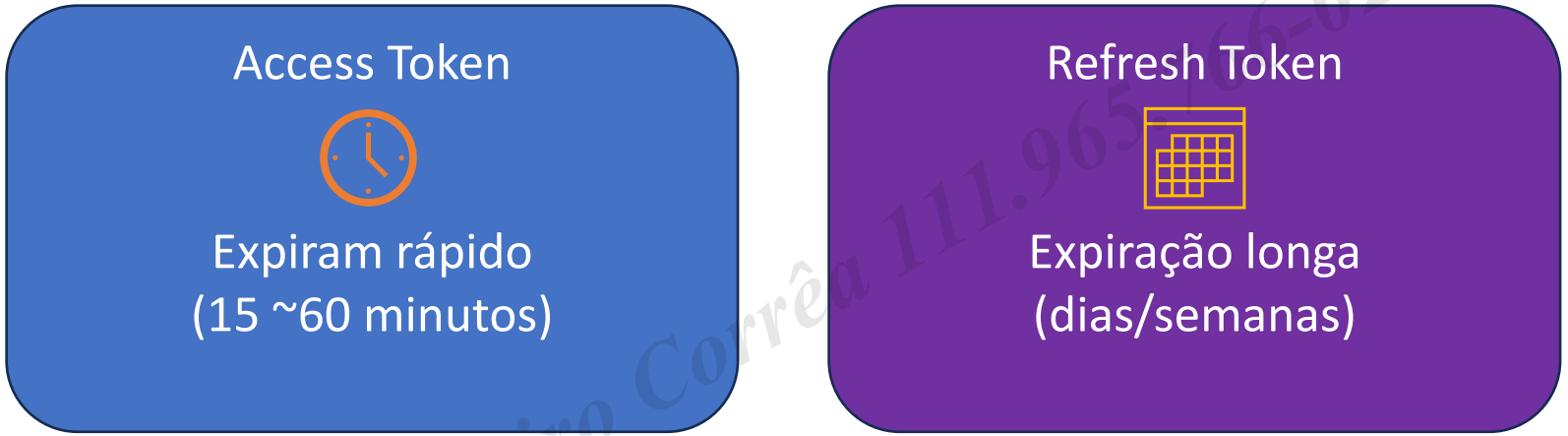


Usada em logins com Google, GitHub, Facebook

JWTs são stateless (sem estado), significando que não é necessário armazenar sessões

Fonte: Adaptado de https://www.youtube.com/watch?v=9JPnN1Z_iSY

Access e Refresh Tokens

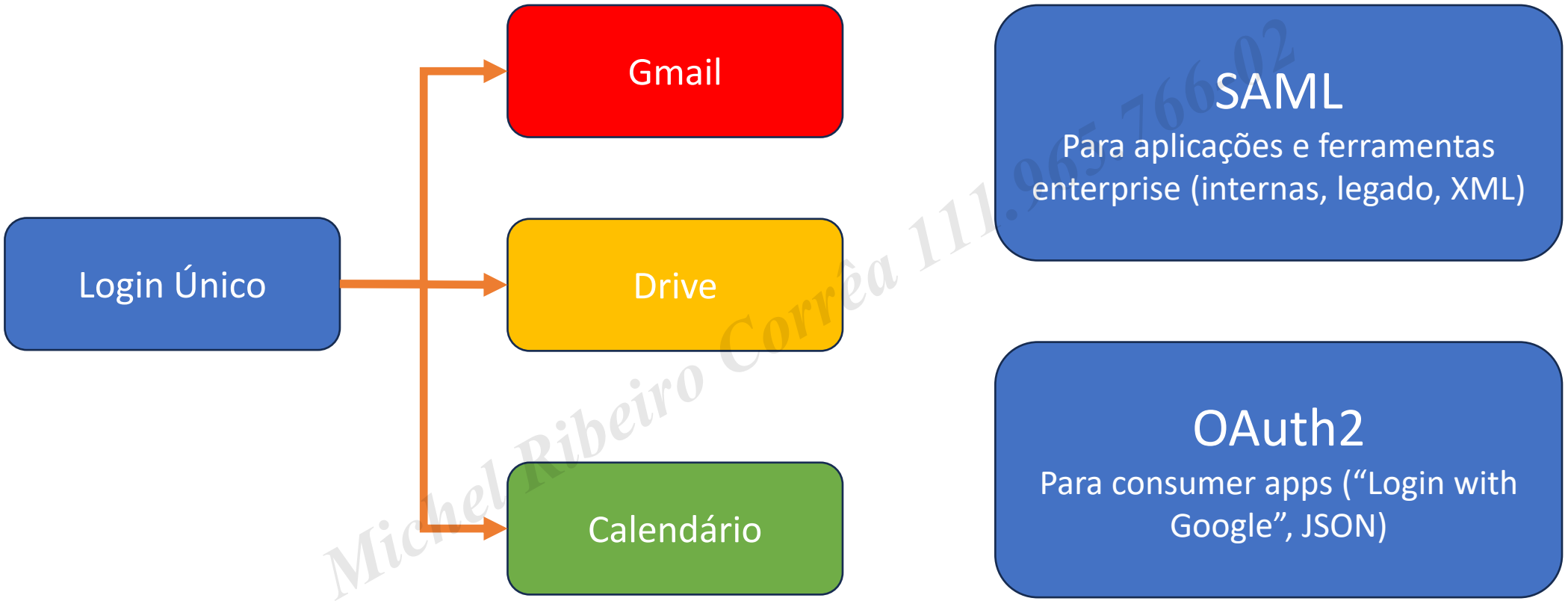


Access Token minimizam o impacto de tokens comprometidos

Refresh Token é uma técnica que serve para que o cliente possa solicitar ao servidor um novo token de autenticação quando o anterior expirar, sem a necessidade de que o usuário se envolva neste processo

Fonte: Adaptado de https://www.youtube.com/watch?v=9JPnN1Z_iSY e https://www.luiztools.com.br/post/implementando-refresh-token-em-node-js/

SSO + Protocolos de Identidade



Protocolos de Identidade definem como aplicações trocam de forma segura informações de login

Fonte: Adaptado de https://www.youtube.com/watch?v=9JPnN1Z_iSY

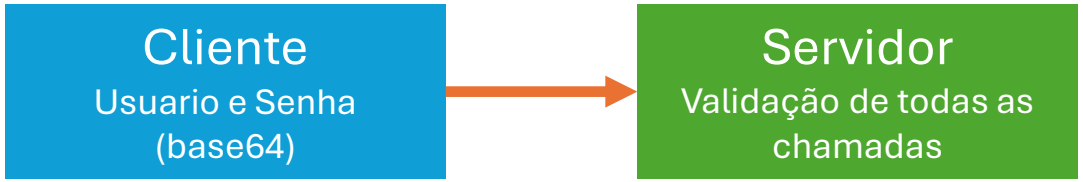
Evolução da Autenticação - Do Básico ao Moderno

- O Basic Authentication, definido na RFC 7617, representa a forma mais simples de autenticação HTTP, onde credenciais são codificadas em Base64 e enviadas em cada requisição. Embora simples de implementar, este método apresenta sérias limitações de segurança, pois as credenciais são transmitidas em cada requisição e podem ser facilmente decodificadas se interceptadas.
- A transição para Session-based Authentication introduziu o conceito de estado no servidor, onde após a autenticação inicial, um identificador de sessão é mantido no servidor e referenciado através de cookies no cliente.
- Este modelo funciona bem em aplicações monolíticas, mas apresenta desafios em microsserviços devido à necessidade de compartilhar estado entre múltiplos serviços.
- O Token-based Authentication representa a evolução natural para arquiteturas distribuídas, onde tokens auto-contidos carregam informações de autenticação e autorização, eliminando a necessidade de estado compartilhado entre serviços.

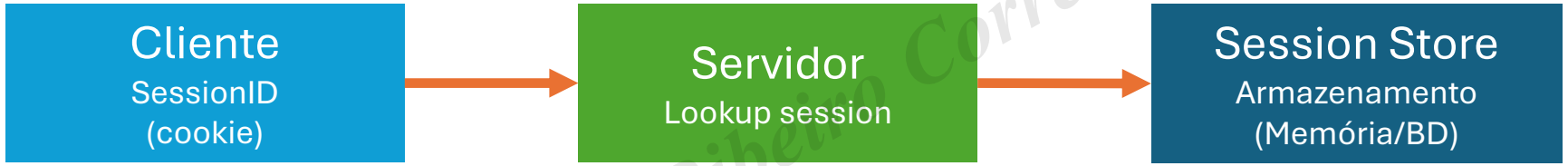
Fonte: RFC 7617 - HTTP Basic Authentication Scheme

Evolução da Autenticação - Do Básico ao Moderno

Autenticação Básica (Anos 1990)



Autenticação baseada em sessão (Anos 2000)



Autenticação baseada em tokens (Anos 2010)

Fonte: RFC 7617 - HTTP Basic Authentication Scheme

OAuth 2.0 Framework - Fundamentos e Arquitetura

- OAuth 2.0 representa um framework de autorização (não autenticação) que permite que aplicações obtenham acesso limitado a recursos de usuários sem expor suas credenciais. O framework define quatro roles principais:
    1. Resource Owner (usuário que possui os dados)
    2. Client (aplicação que deseja acessar os dados)
    3. Authorization Server (servidor que autentica o usuário e emite tokens)
    4. Resource Server (servidor que hospeda os recursos protegidos).
- Esta separação de responsabilidades permite implementações flexíveis e seguras em arquiteturas distribuídas.

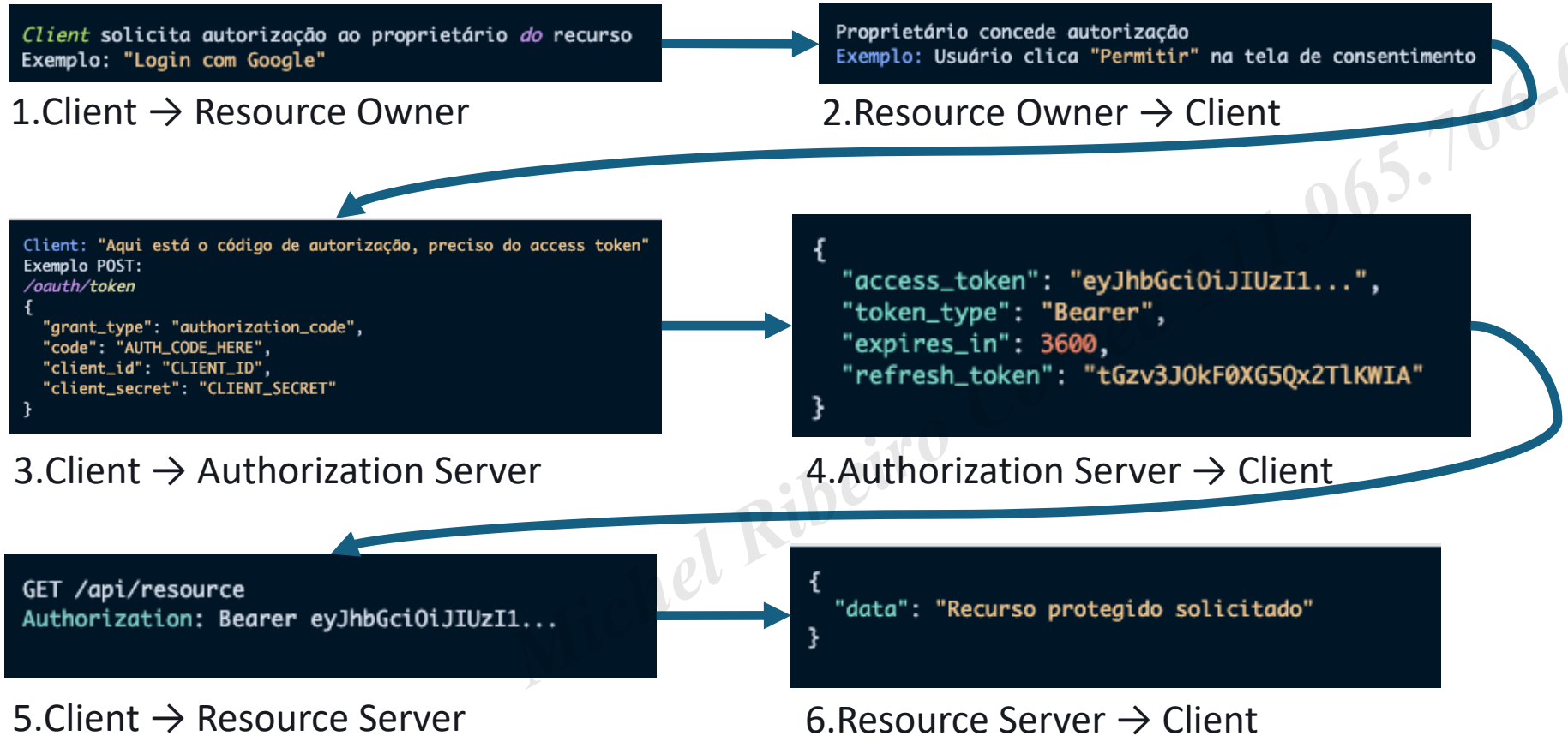
Fonte: OWASP API Security Top 10

OAuth 2.0 Framework - Fundamentos e Arquitetura

- O OAuth 2.0 tornou-se o padrão de facto para autorização em APIs modernas devido à sua flexibilidade e capacidade de suportar diferentes tipos de aplicações, desde aplicações web tradicionais até aplicações móveis e IoT.
- O framework suporta múltiplos grant types (fluxos de autorização) que podem ser escolhidos baseados no tipo de aplicação e requisitos de segurança.
- A implementação adequada do OAuth 2.0 em microsserviços requer cuidadosa consideração de aspectos como token validation, scope management e revocation mechanisms.

Fonte: OWASP API Security Top 10

OAuth 2.0 Framework



Fluxo:

1. O cliente solicita autorização ao Proprietário do Recurso

2. O Proprietário do Recurso concede autorização

3. O cliente solicita o token de acesso ao Servidor de Autorização

4. O Servidor de Autorização valida e emite o token de acesso

5. O cliente usa o token de acesso para acessar o Servidor de Recurso

6. O Servidor de Recurso valida o token e fornece o recurso

Fonte: Fonte: RFC 6749 - OAuth 2.0 Authorization Framework e exemplos e diagrama criados por GenAI via Amazon Q Developer CLI

OAuth 2.0 Grant Types - Fluxos de Autorização

- Os diferentes grant types do OAuth 2.0 foram projetados para atender cenários específicos de aplicação, cada um com suas próprias características de segurança e casos de uso.

Authorization Code Grant	Client Credentials Grant	Resource Owner Password Credentials Grant	Implicit Grant	Device Authorization Grant
• É o mais seguro e amplamente utilizado para aplicações web tradicionais, onde o código de autorização é trocado por um access token em uma comunicação server-to-server segura.	• É ideal para comunicação service-to-service em microsserviços, onde não há um usuário final envolvido na transação.	• Deve ser usado apenas em cenários onde existe alta confiança entre o cliente e o servidor de autorização, como aplicações first-party.	• Originalmente projetado para Single Page Applications, foi descontinuado devido a vulnerabilidades de segurança e substituído pelo Authorization Code Grant com PKCE.	• Atende cenários específicos como smart TVs e dispositivos IoT que têm capacidades limitadas de entrada de dados.

Fonte: Fonte: RFC 6749 - OAuth 2.0 Authorization Framework

JWT (JSON Web Tokens) - Estrutura e Implementação

- JSON Web Tokens representam uma forma compacta e auto contida de transmitir informações entre partes de forma segura. A estrutura de um JWT consiste em três partes separadas por pontos:
    1. Header (contém metadados sobre o token)
    2. Payload (contém as claims ou declarações sobre o usuário)
    3. Signature (garante a integridade do token).
- Esta estrutura permite que microsserviços validem tokens localmente sem necessidade de consultar um servidor central, melhorando significativamente a performance e escalabilidade.

Fonte: RFC 7519 - JSON Web Token (JWT)

JWT (JSON Web Tokens) - Estrutura e Implementação

O token é criptografado ?

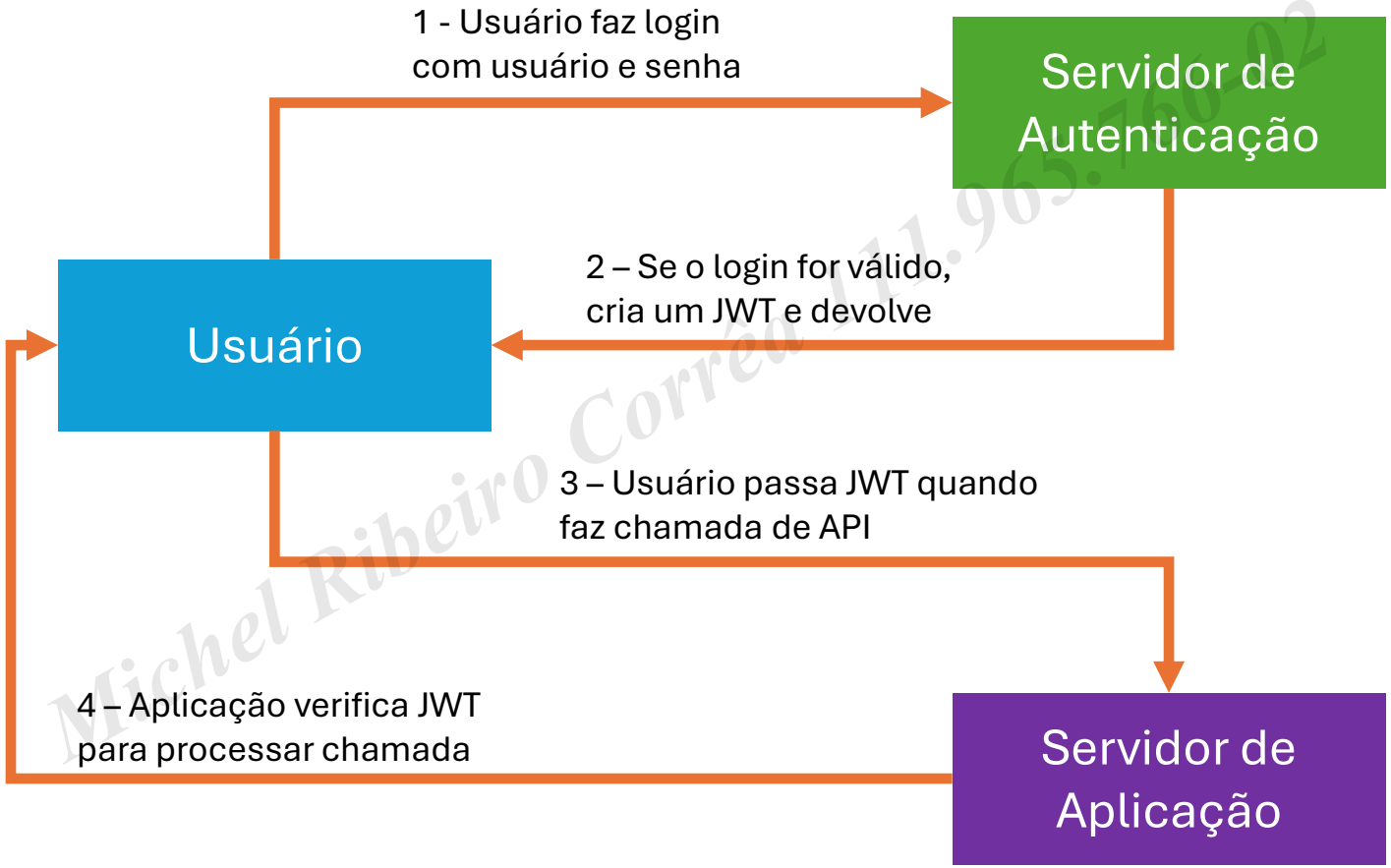
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjEsImlhdCI6MTc1MTY1NDY3MSwiZXhwIjoxNzUxNjU0OTcxfQ.0bbF7setIFnWvSuOKGfir32O9GLS9Fma-z91hq9uSQU
```

O token apenas é codificado em base64, uma representação textual de um conjunto de bytes, separada em três partes por pontos

https://www.base64decode.org/

Fonte: Adaptado de https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs-2/

JWT (JSON Web Tokens) - Estrutura e Implementação



Fonte: Adaptado de https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/

JWT - Vantagens, Desvantagens e Melhores Práticas

- A implementação de JWTs em microsserviços oferece vantagens significativas, incluindo natureza stateless que elimina a necessidade de armazenamento de sessão compartilhado, compatibilidade cross-domain que facilita integração entre diferentes serviços e domínios, e capacidade de carregar informações estruturadas que reduzem a necessidade de consultas adicionais a serviços de identidade.
- A portabilidade dos JWTs permite que sejam utilizados em diferentes plataformas e linguagens de programação sem modificações.
- No entanto, JWTs também apresentam desvantagens importantes que devem ser consideradas na arquitetura:
    1. O tamanho maior comparado a session IDs tradicionais pode impactar performance, especialmente em aplicações com muitas requisições.
    2. A complexidade de revogação é um desafio significativo, pois tokens são válidos até sua expiração natural, tornando difícil invalidar tokens comprometidos imediatamente.
    3. A exposição de dados sensíveis no payload pode representar riscos de segurança se o token for interceptado, mesmo que não possa ser alterado devido à assinatura digital.

Fonte: Adaptado de https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/

Access e Refresh Tokens - Gerenciamento de Ciclo de Vida

- O padrão de Access e Refresh Tokens representa uma abordagem sofisticada para balancear segurança e usabilidade em sistemas de autenticação.

1. Access tokens são projetados para ter vida curta (tipicamente 15-30 minutos) e são utilizados para acessar recursos protegidos
2. Refresh tokens têm vida mais longa (dias ou semanas) e são utilizados exclusivamente para obter novos access tokens.

- Esta separação permite que sistemas mantenham segurança alta com impacto mínimo na experiência do usuário.
- A implementação de token rotation adiciona uma camada extra de segurança, onde cada uso de um refresh token resulta na emissão de um novo par de access/refresh tokens, invalidando os tokens anteriores.
- Este mecanismo torna mais difícil para atacantes utilizarem tokens roubados por períodos prolongados.
- Em microsserviços, a revogação de tokens deve ser coordenada através de múltiplos serviços, frequentemente implementada através de token blacklists distribuídas ou consultas em tempo real a serviços de validação centralizados.

Fonte: Adaptado de https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/

Single Sign-On (SSO) - Implementação em microsserviços

- Single Sign-On em microsserviços representa um desafio arquitetural complexo que requer coordenação cuidadosa entre múltiplos serviços e domínios. O conceito fundamental é permitir que usuários se autentiquem uma vez em um Identity Provider (IdP) centralizado e então acessem múltiplas aplicações e serviços sem necessidade de re-autenticação.
- Em arquiteturas de microsserviços, isso significa que cada serviço deve ser capaz de validar tokens emitidos pelo IdP central e extrair informações de identidade necessárias para suas operações específicas.
- A implementação de SSO oferece benefícios significativos incluindo experiência do usuário melhorada (eliminando múltiplos logins), gestão centralizada de identidades e políticas de segurança, e redução de surface de ataque através da centralização da autenticação.
- No entanto, também introduz riscos como ponto único de falha (se o IdP ficar indisponível, todos os serviços são afetados) e complexidade adicional na implementação de logout global, onde terminar uma sessão deve invalidar acesso a todos os serviços conectados.

Fonte: https://aws.amazon.com/what-is/sso/

SAML 2.0 - Enterprise Identity Federation

- Security Assertion Markup Language (SAML) 2.0 representa o padrão maduro para federação de identidades em ambientes corporativos, especialmente em integrações com sistemas legados como Active Directory. SAML utiliza XML para estruturar assertions (declarações) sobre usuários, incluindo informações de autenticação, autorização e atributos.
- Em microsserviços, SAML é frequentemente utilizado como ponte entre sistemas corporativos tradicionais e arquiteturas modernas, permitindo que funcionários utilizem suas credenciais corporativas para acessar novos serviços.
- O protocolo SAML suporta dois fluxos principais: IdP-initiated (onde o processo começa no Identity Provider) e SP-initiated (onde o processo começa no Service Provider). As assertions SAML são assinadas digitalmente e podem ser criptografadas, garantindo integridade e confidencialidade das informações de identidade.
- Em arquiteturas de microsserviços, SAML assertions são frequentemente convertidas para formatos mais modernos como JWT para facilitar o processamento por serviços individuais, mantendo compatibilidade com sistemas corporativos existentes.

Fonte: https://wiki.oasis-open.org/security

OpenID Connect (OIDC) - Identidade sobre OAuth 2.0

- OpenID Connect representa a evolução moderna dos protocolos de identidade, construído como uma camada de identidade sobre o framework OAuth 2.0.
- Enquanto OAuth 2.0 foca em autorização, OIDC adiciona capacidades de autenticação através de ID Tokens padronizados em formato JWT.
- Esta abordagem combina a flexibilidade e maturidade do OAuth 2.0 com informações estruturadas de identidade, tornando-se ideal para implementações em microsserviços que precisam de informações detalhadas sobre usuários.
- O protocolo OIDC introduz conceitos importantes como ID Token (contém informações sobre o evento de autenticação e o usuário), UserInfo endpoint (fornece informações adicionais sobre o usuário) e Discovery endpoint (permite configuração automática de clientes).
- A capacidade de discovery é particularmente valiosa em microsserviços, pois permite que serviços se configurem automaticamente para trabalhar com diferentes provedores de identidade sem necessidade de configuração manual extensiva.

Fonte: https://openid.net/developers/how-connect-works/

Exercício

- Demonstração prática de autenticação e autorização em microsserviços usando OAuth 2.0 e JWT tokens

1. OAuth 2.0 Grant Types

- Authorization Code Flow: Para usuários web
- Client Credentials Flow: Para comunicação service-to-service
- Refresh Token Rotation: Renovação segura de tokens

2. JWT (JSON Web Tokens)

- Access Tokens: Curta duração (15 min)
- Refresh Tokens: Longa duração (7 dias)
- Claims customizadas: user_id, role, scopes

Autorização e Controle de Acesso

Parte 3

Role-Based Access Control (RBAC) - Controle Baseado em Papéis

- Role-Based Access Control representa o modelo de autorização mais amplamente adotado em sistemas corporativos, onde permissões são agrupadas em roles (papéis) que são então atribuídos a usuários.
- Em microsserviços, RBAC oferece simplicidade de gestão através da centralização de definições de papéis, permitindo que administradores gerenciem permissões em um nível mais alto de abstração. Cada role representa um conjunto de permissões relacionadas a funções específicas dentro da organização, como "Administrador", "Editor" ou "Visualizador".
- A implementação de RBAC em microsserviços pode seguir diferentes padrões:
- centralizado (onde roles são definidos globalmente)
- distribuído (onde cada serviço define seus próprios roles).
- A hierarquia de roles permite herança de permissões, onde roles mais específicos herdam permissões de roles mais gerais. No entanto, RBAC apresenta limitações em cenários que requerem controle de acesso mais granular ou baseado em contexto, pois as decisões são baseadas apenas na identidade do usuário e seus roles, não considerando fatores como localização, horário ou atributos específicos dos recursos.

Fonte: OWASP API Security Top 10

Role-Based Access Control (RBAC) - Controle Baseado em Papéis

Recurso/Ação	Admin	Viewer	Customer
Products Read	✅	✅	✅
Products Write	✅	❌	❌
Orders Read Own	✅	❌	✅
Orders Read All	✅	❌	❌
Cart Management	❌	❌	✅
Sales Reports	✅	✅	❌
User Management	✅	❌	❌

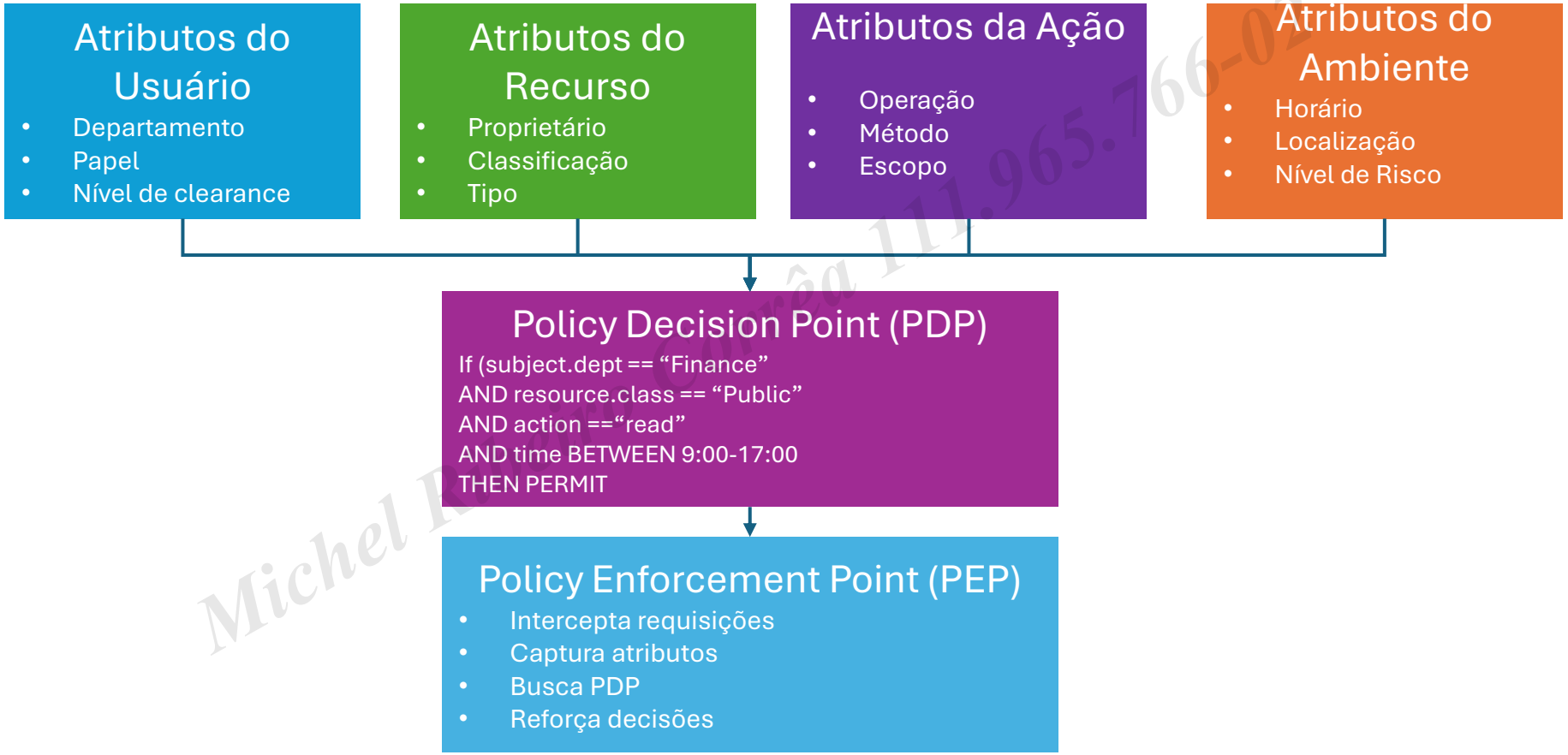
Fonte: NIST RBAC Model - ANSI INCITS 359-2004 e diagrama criado por GenAI via Amazon Q Developer CLI

Attribute-Based Access Control (ABAC) - Controle Baseado em Atributos

- Attribute-Based Access Control representa a evolução natural do controle de acesso para ambientes complexos e dinâmicos como microsserviços. ABAC tem como características:
    1. decisões de autorização baseadas em atributos do usuário (departamento, nível de clearance, localização)
    2. recurso (classificação, proprietário, tipo)
    3. ação (leitura, escrita, execução)
    4. ambiente (horário, localização, nível de risco)
- Esta abordagem oferece granularidade extrema e flexibilidade para implementar políticas de segurança sofisticadas que consideram contexto dinâmico.
- Em microsserviços, ABAC permite que cada serviço implemente políticas específicas baseadas em seus requisitos únicos, enquanto mantém consistência através de um framework comum de avaliação de atributos.
- A implementação típica envolve um Policy Decision Point (PDP) que avalia políticas e um Policy Enforcement Point (PEP) que executa as decisões. O desafio principal está na complexidade de gestão e performance, pois cada decisão de autorização pode requerer avaliação de múltiplos atributos e políticas complexas.

Fonte: NIST SP 800-162 - Guide to Attribute Based Access Control

Attribute-Based Access Control (ABAC) - Controle Baseado em Atributos



Fonte: NIST SP 800-162 - Guide to Attribute Based Access Control

Policy-Based Access Control (PBAC) - Controle Baseado em Políticas

- Policy-Based Access Control representa uma abordagem onde decisões de autorização são tomadas através da avaliação de políticas expressas em linguagens específicas como XACML (eXtensible Access Control Markup Language). PBAC separa claramente a lógica de autorização da implementação dos serviços, permitindo que políticas sejam definidas, modificadas e auditadas independentemente do código da aplicação.
- Esta separação é particularmente valiosa em microsserviços, onde políticas podem ser aplicadas consistentemente através de múltiplos serviços. A arquitetura PBAC típica inclui:
- Policy Administration Point (PAP) para criação e gestão de políticas
- Policy Decision Point (PDP) para avaliação de políticas
- Policy Enforcement Point (PEP) para execução de decisões
- Policy Information Point (PIP) para fornecimento de informações contextuais.
- Em microsserviços, cada serviço pode atuar como PEP, consultando um PDP centralizado ou distribuído para decisões de autorização. A linguagem XACML permite expressar políticas complexas incluindo obrigações (ações que devem ser executadas junto com a decisão) e advice (recomendações opcionais).

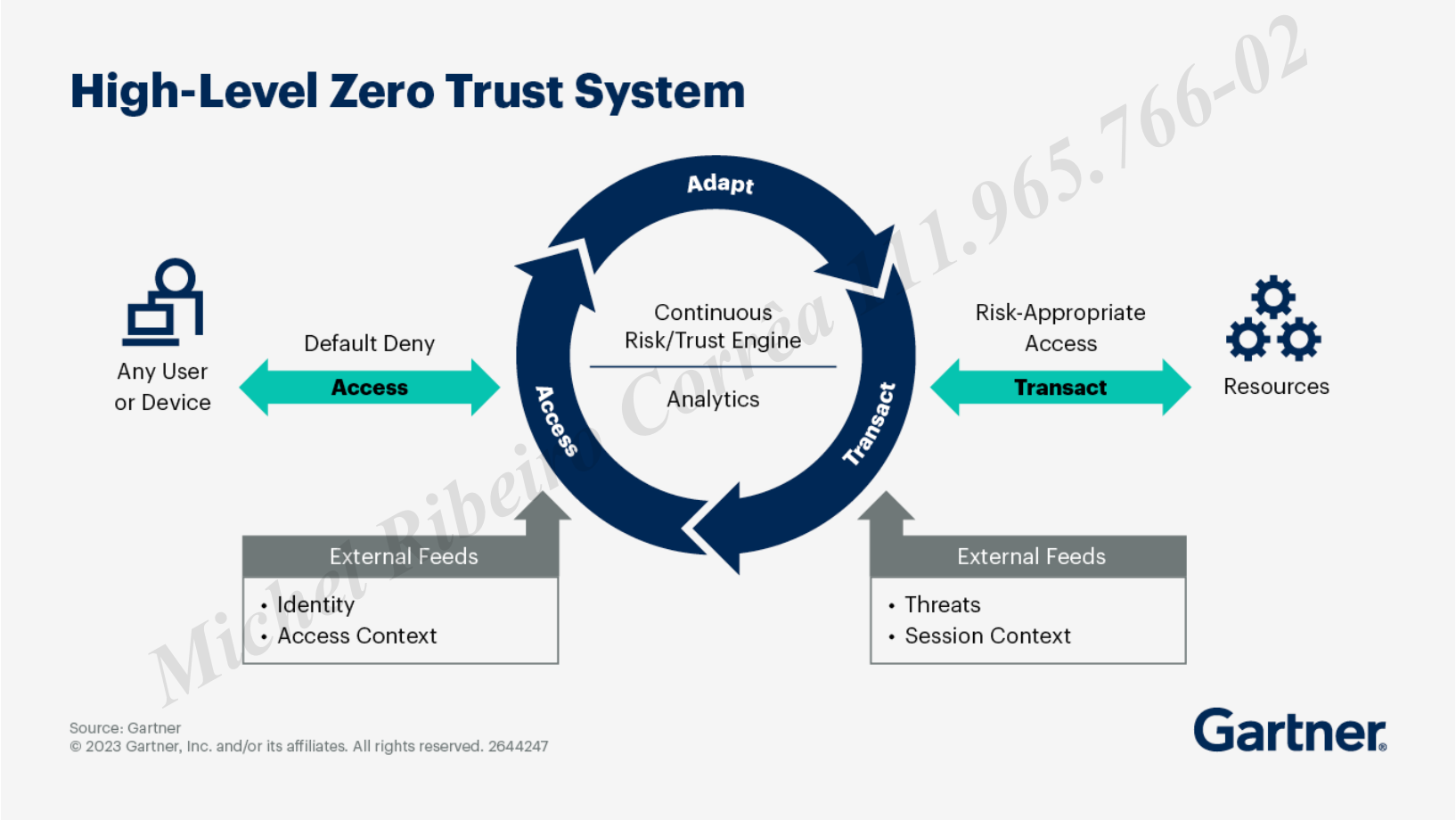
Fonte: OASIS XACML 3.0 Core Specification

Zero Trust Architecture - "Never Trust, Always Verify"

- Zero Trust Architecture representa uma mudança fundamental na abordagem de segurança, abandonando o conceito tradicional de perímetro de rede confiável em favor de um modelo onde nenhuma entidade (usuário, dispositivo ou serviço) é considerada confiável por padrão.
- Em microsserviços, Zero Trust é particularmente relevante pois cada serviço deve validar independentemente a identidade e autorização de cada requisição, independentemente de sua origem. O princípio "never trust, always verify" significa que mesmo comunicações internas entre serviços devem ser autenticadas e autorizadas.
- A implementação de Zero Trust em microsserviços envolve múltiplas camadas:
    1. Micro-segmentação de rede para isolar serviços
    2. Continuous verification onde cada transação é validada
    3. Least privilege access onde cada entidade recebe apenas as permissões mínimas necessárias, e assume breach mentality onde o sistema é projetado assumindo que violações de segurança irão ocorrer
    4. Service mesh technologies como Istio facilitam a implementação de Zero Trust através de mutual TLS automático, políticas de rede granulares e observabilidade de segurança.

Fonte: NIST SP 800-207 - Zero Trust Architecture

Zero Trust Architecture - "Never Trust, Always Verify"



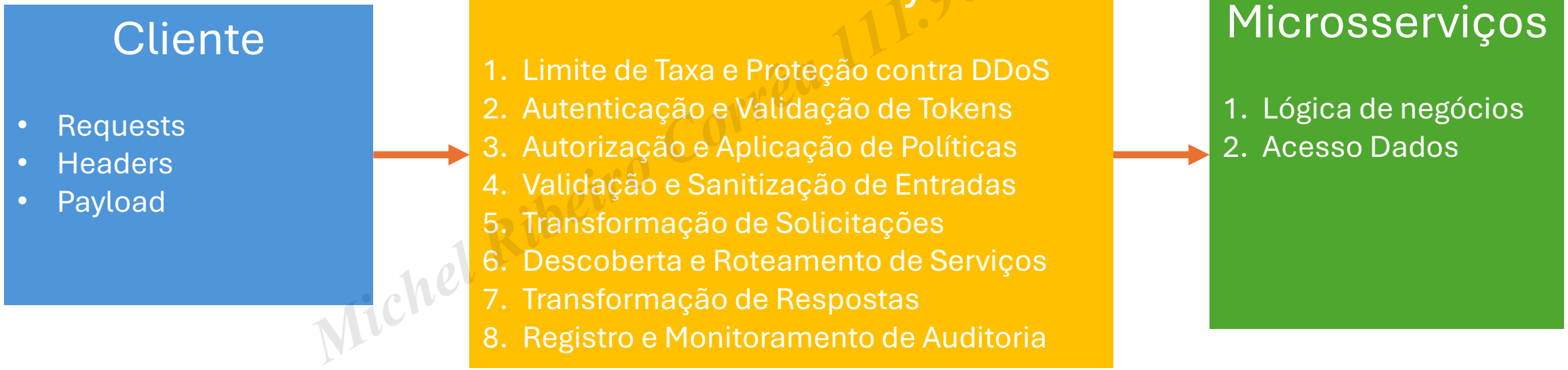
Fonte: https://www.gartner.com/en/industries/government-public-sector/topics/zero-trust

API Gateway - Ponto Central de Controle de Segurança

- O API Gateway em arquiteturas de microsserviços serve como o ponto central de enforcement de políticas de segurança, atuando como um proxy reverso que intercepta todas as requisições antes que alcancem os serviços internos.
- Esta posição estratégica permite implementar múltiplas camadas de segurança incluindo autenticação, autorização, rate limiting, validação de input e transformação de requisições/respostas.
- O API Gateway também facilita a implementação de padrões de segurança consistentes através de toda a arquitetura, eliminando a necessidade de duplicar lógica de segurança em cada microserviço.
- Além das funcionalidades básicas de segurança, API Gateways modernos oferecem capacidades avançadas como adaptive authentication (ajustando requisitos de autenticação baseados em risco), threat detection (identificando padrões suspeitos de tráfego), e analytics de segurança (fornecendo visibilidade sobre tentativas de acesso e ataques).
- A implementação de circuit breakers e bulkhead patterns no API Gateway ajuda a prevenir que ataques ou falhas se propaguem através da arquitetura, mantendo a disponibilidade dos serviços críticos.

Fonte: Gartner API Management Market Guide

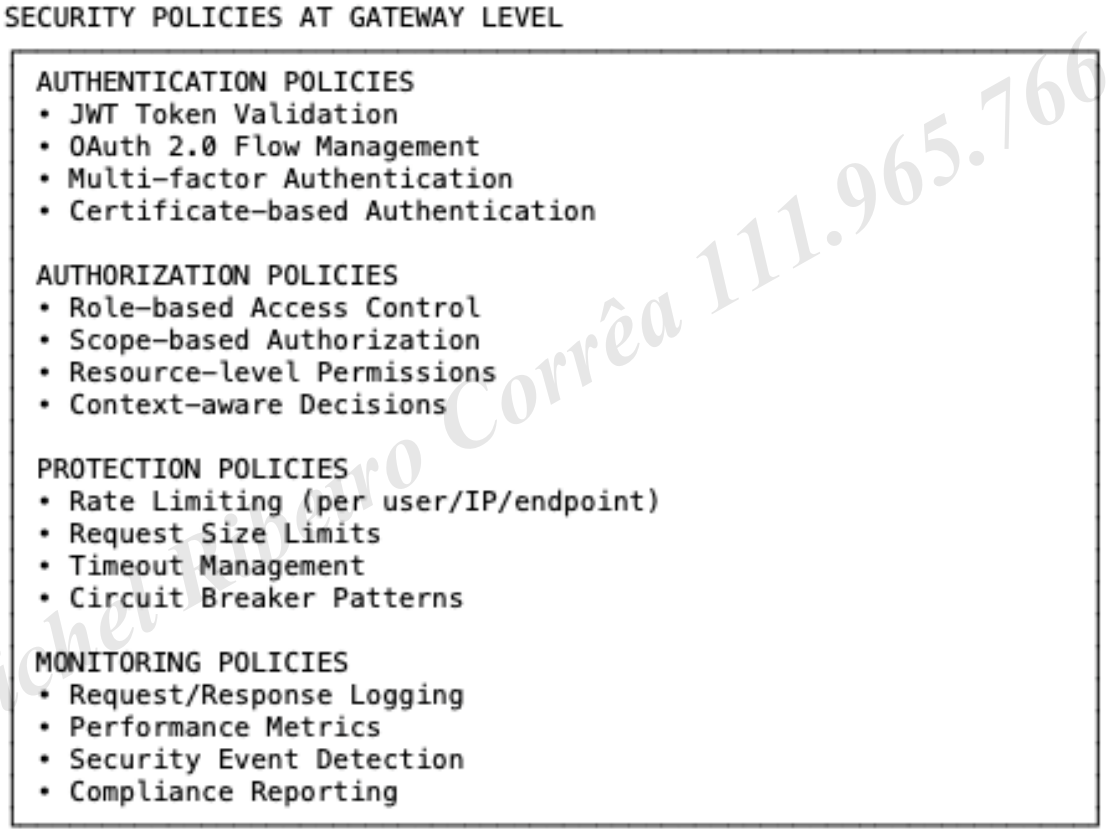
API Gateway - Ponto Central de Controle de Segurança



Fluxo de solicitação através do gateway de API

Fonte: Gartner API Management Market Guide

API Gateway - Ponto Central de Controle de Segurança



Fonte: Gartner API Management Market Guide e diagrama criado por GenAI via Amazon Q Developer CLI

Service Mesh - Segurança na Camada de Infraestrutura

- Service Mesh representa uma camada de infraestrutura dedicada que gerencia comunicação service-to-service, oferecendo capacidades avançadas de segurança sem necessidade de modificações no código das aplicações.
- Em termos de segurança, service mesh implementa **mutual TLS (mTLS) automático** entre todos os serviços, garantindo que toda comunicação interna seja criptografada e autenticada. Esta abordagem elimina a necessidade de implementar TLS manualmente em cada serviço, reduzindo significativamente a complexidade e possibilidade de erros de configuração.
- O service mesh também implementa identity-based networking, onde políticas de rede são definidas baseadas na identidade dos serviços em vez de endereços IP, tornando as políticas mais resilientes a mudanças na infraestrutura.
- A observabilidade de segurança fornecida pelo service mesh inclui métricas detalhadas sobre tentativas de conexão, falhas de autenticação e violações de políticas.

Fonte: OWASP API Security Top 10

Exercício

- Demonstração prática de controle de acesso baseado em roles (RBAC)
- **Autorização RBAC**
- Admin: CRUD completo, relatórios, gestão de usuários
- Viewer: Visualização e relatórios de vendas
- Customer: Navegação, compras, carrinho

Soluções de Nuvem Pública

Parte 4

AWS Identity Services - Cognito e IAM

- A AWS Amazon Web Services oferece um conjunto abrangente de serviços de identidade e acesso que se integram naturalmente com arquiteturas de microsserviços.
- O Amazon Cognito fornece duas funcionalidades principais: User Pools para autenticação de usuários finais e Identity Pools para acesso federado a recursos AWS. User Pools suportam múltiplos provedores de identidade social (Google, Facebook, Amazon), SAML e OpenID Connect, além de oferecer recursos avançados como adaptive authentication que ajusta requisitos de segurança baseados em risco detectado.
- O AWS Identity and Access Management (IAM) forma a base do controle de acesso em toda a plataforma AWS, permitindo definição granular de permissões através de policies JSON.
- Em microsserviços rodando em AWS, IAM roles podem ser assumidos por serviços para acessar recursos específicos sem necessidade de credenciais hardcoded.
- O Security Token Service (STS) permite geração de credenciais temporárias com escopo limitado, implementando o princípio de least privilege. A integração entre Cognito e IAM permite que usuários autenticados assumam roles específicos baseados em seus atributos e grupos.

Fonte: AWS Cognito Developer Guide

Azure Identity Platform - Azure AD e B2C

- Microsoft Azure oferece uma plataforma de identidade robusta centrada no Azure Active Directory (Azure AD), que serve tanto para identidades corporativas quanto para aplicações voltadas ao consumidor. O Azure AD fornece capacidades avançadas de Conditional Access que permitem implementar políticas de acesso baseadas em risco, localização, dispositivo e comportamento do usuário.
- Para microsserviços, Azure AD oferece integração nativa com OAuth 2.0 e OpenID Connect, além de suporte para SAML em cenários de federação corporativa.
- O Azure AD B2C (Business-to-Consumer) é especificamente projetado para aplicações que atendem consumidores finais, oferecendo customização completa da experiência de usuário através de custom policies e user journeys.
- B2C suporta múltiplos provedores de identidade social e permite implementação de fluxos de autenticação complexos incluindo verificação de email, recuperação de senha e autenticação multi-fator.
- O Azure Key Vault complementa a plataforma fornecendo gerenciamento seguro de secrets, chaves criptográficas e certificados utilizados pelos microsserviços.

Fonte: Microsoft Azure AD Documentation

Google Cloud Identity Platform - Firebase Auth Evolution

- Google Cloud Platform oferece uma plataforma de identidade moderna através do Identity Platform, que representa a evolução do Firebase Authentication para cenários empresariais.
- O Identity Platform suporta múltiplos provedores de identidade incluindo Google, Facebook, Twitter, GitHub, além de provedores SAML e OpenID Connect personalizados. Para microsserviços, oferece SDKs em múltiplas linguagens e APIs REST que facilitam integração com diferentes tecnologias e frameworks.
- O Google Cloud IAM (Identity and Access Management) implementa um modelo hierárquico de permissões baseado na estrutura organizacional do Google Cloud (Organization > Folder > Project > Resource). Este modelo permite herança de permissões e implementação granular de controles de acesso.
- Service accounts no GCP são identidades especiais para aplicações e microsserviços, permitindo autenticação service-to-service sem necessidade de credenciais de usuário.
- O Cloud Identity complementa a plataforma oferecendo gerenciamento unificado de identidades corporativas com sincronização LDAP e integração com Google Workspace.

Fonte: Google Cloud Identity Platform Documentation

SECURITY IMPLEMENTATION CHECKLIST

AUTHENTICATION & AUTHORIZATION	COMMUNICATION SECURITY	INPUT VALIDATION & PROTECTION
☐ Implement OAuth 2.0 with PKCE	☐ Enforce HTTPS everywhere	☐ Validate input at gateway and service levels
☐ Use short-lived access tokens (15-30 min)	☐ Implement mTLS for service-to-service	☐ Implement rate limiting (multiple layers)
☐ Implement refresh token rotation	☐ Use certificate-based authentication	☐ Use request size limits
☐ Validate all JWT claims (iss, aud, exp, nbf)	☐ Implement proper certificate lifecycle mgmt	☐ Implement timeout management
☐ Use strong signing algorithms (RS256, ES256)		☐ Deploy DDoS protection
☐ Implement proper key rotation		

SECRETS & CONFIGURATION	MONITORING & OBSERVABILITY
☐ Use dedicated secret management solutions	☐ Implement comprehensive audit logging
☐ Implement secret rotation	☐ Use correlation IDs for tracing
☐ Avoid hardcoded credentials	☐ Monitor security metrics and KPIs
☐ Use environment-specific configurations	☐ Implement anomaly detection
	☐ Set up security alerting

Fonte: OWASP Microservices Security Verification Standard

Monitoramento e Observabilidade de Segurança

- A observabilidade de segurança em microsserviços vai além do logging tradicional, requerendo uma abordagem holística que combine métricas, traces e logs para fornecer visibilidade completa sobre o estado de segurança da arquitetura.
- Security metrics devem incluir KPIs como taxa de tentativas de autenticação falhadas, latência de validação de tokens, distribuição de acessos por usuário/serviço, e padrões de tráfego anômalos. A implementação de distributed tracing permite rastrear requisições de segurança através de múltiplos serviços, identificando gargalos e pontos de falha.
- A detecção de anomalias deve utilizar machine learning para identificar padrões suspeitos como acessos fora do horário normal, tentativas de acesso a recursos não autorizados, ou padrões de tráfego indicativos de ataques automatizados.
- A integração com SIEM (Security Information and Event Management) permite correlação de eventos de segurança com contexto mais amplo da infraestrutura. Incident response automation deve incluir playbooks para cenários comuns como tokens comprometidos, tentativas de brute force, ou violações de políticas de acesso.

Fonte: NIST Cybersecurity Framework - Detect Function

Referências Bibliográficas e Recursos

Parte 4

Referências Bibliográficas

- "Microservices Security in Action" por Prabath Siriwardena e Nuwan Dias oferece uma abordagem prática e abrangente para implementação de segurança em arquiteturas de microsserviços, cobrindo desde conceitos básicos até implementações avançadas com OAuth 2.0, JWT e service mesh
- "OAuth 2 in Action" por Justin Richer e Antonio Sanso fornece entendimento profundo do framework OAuth 2.0 com exemplos práticos e cenários reais de implementação.
- "Identity and Data Security for Web Development" por Jonathan LeBlanc e Tim Messerschmidt aborda aspectos práticos de segurança em desenvolvimento web moderno, incluindo melhores práticas para autenticação e proteção de dados.
- "Zero Trust Networks" por Evan Gilman e Doug Barth apresenta os princípios e implementação prática de arquiteturas Zero Trust.
- "API Security in Action" por Neil Madden oferece guia completo para proteção de APIs, incluindo técnicas avançadas de autenticação e autorização.

Especificações e Padrões Técnicos

- RFC 6749 (OAuth 2.0 Authorization Framework) define o framework fundamental para autorização em APIs modernas: https://datatracker.ietf.org/doc/html/rfc6749
- RFC 7519 (JSON Web Token) especifica o formato e uso de JWTs para transmissão segura de informações: https://datatracker.ietf.org/doc/html/rfc7519
- RFC 8252 (OAuth 2.0 for Native Apps) fornece diretrizes específicas para implementação segura em aplicações móveis e nativas: https://datatracker.ietf.org/doc/html/rfc8252
- OpenID Connect Core 1.0 adiciona camada de identidade sobre OAuth 2.0, permitindo autenticação padronizada: https://openid.net/specs/openid-connect-core-1_0-final.html
- NIST SP 800-207 (Zero Trust Architecture) define princípios e arquiteturas para implementação de Zero Trust: https://csrc.nist.gov/pubs/sp/800/207/final
- OASIS SAML 2.0 especifica federação de identidades para ambientes corporativos: https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html
- NIST SP 800-63 (Digital Identity Guidelines) fornece framework abrangente para gestão de identidades digitais: https://csrc.nist.gov/publications/sp800

Whitepapers e Recursos dos Provedores de Nuvem

- AWS Security Best Practices: https://docs.aws.amazon.com/whitepapers/latest/aws-security-best-practices/welcome.html
- AWS Well-Architected: https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html
- AWS Identity Federation: https://aws.amazon.com/pt/identity/federation/
- Microservices Security on AWS: https://docs.aws.amazon.com/whitepapers/latest/microservices-on-aws/microservices-on-aws.html
- Microsoft Azure Security and Compliance Blueprint: https://learn.microsoft.com/pt-br/azure/governance/blueprints/samples/
- Zero Trust Guidance Center: https://learn.microsoft.com/en-us/security/zero-trust/
- Identity and Access Management for Azure: https://azure.microsoft.com/pt-br/products/category/identity
- Google Cloud Security Foundations Guide: https://cloud.google.com/architecture/blueprints/security-foundations
- BeyondCorp: A New Approach to Enterprise: https://research.google/pubs/beyondcorp-a-new-approach-to-enterprise-security/

Recursos Adicionais e Comunidades

- OWASP (Open Web Application Security Project) fornece recursos essenciais incluindo Top 10, Cheat Sheets e Testing Guide, com foco específico em segurança de APIs e microsserviços: https://owasp.org/
- NIST Cybersecurity Framework oferece abordagem estruturada para gestão de riscos de segurança: https://www.nist.gov/cyberframework
- Cloud Security Alliance (CSA) desenvolve melhores práticas para segurança em nuvem: https://cloudsecurityalliance.org/
- SANS Institute oferece treinamentos especializados e pesquisas em segurança de aplicações: https://www.sans.org/
- Auth0 Blog publica artigos técnicos regulares sobre identidade e segurança: https://auth0.com/blog/
- OAuth Working Group (IETF): https://datatracker.ietf.org/wg/oauth/about/
- OpenID Foundation: https://openid.net/foundation/
- Cloud Native Computing Foundation (CNCF) desenvolve padrões para segurança em ambientes cloud-native: https://www.cncf.io/

Obrigado!

Prof. José Maria Cesário Jr. | https://www.linkedin.com/in/josemariacesariojunior/

MBA USP ESALQ