Semana 11 - Aula 1

Tópico Principal da Aula: Segurança em aplicações back-end

Subtítulo/Tema Específico: Métodos de autenticação

Código da aula: [SIS]ANO2C2B2S11A1

Objetivos da Aula:

- Implementar diferentes métodos de autenticação.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Slide 05 - Objetivos da aula

- **Definição:** A aula tem como principal objetivo capacitar os estudantes a implementar diversos métodos de autenticação em aplicações back-end. A autenticação é o processo de verificar a identidade de um usuário ou sistema, garantindo que apenas entidades legítimas tenham acesso a recursos protegidos.
- **Aprofundamento/Complemento (se necessário):** Em um cenário de desenvolvimento back-end, a autenticação é a primeira linha de defesa contra acessos não autorizados. Ela é fundamental para a segurança dos dados e da integridade da aplicação. Existem diferentes estratégias e protocolos de autenticação, cada um com suas características e casos de uso, como autenticação baseada em senha, tokens, biometria, entre outros. A escolha do método mais adequado depende dos requisitos de segurança, usabilidade e do ambiente tecnológico da aplicação.
- **Exemplo Prático:** Ao desenvolver um sistema de login para um e-commerce, o backend precisa autenticar o usuário que tenta acessar sua conta. Isso pode ser feito verificando o nome de usuário e a senha fornecidos contra um banco de dados de usuários registrados. Se as credenciais estiverem corretas, o sistema autentica o usuário e permite o acesso.

Referência do Slide: Slide 06 - Habilidades socioemocionais e atividades

- **Definição:** Este slide destaca as habilidades socioemocionais trabalhadas na aula, como a curiosidade e a resiliência na resolução de problemas computacionais, e as atividades propostas para o desenvolvimento do conteúdo, incluindo a exposição, atividade prática e a seção "Então, ficamos assim...".

- **Aprofundamento/Complemento (se necessário):** Além das habilidades técnicas, o desenvolvimento de software exige um conjunto de habilidades socioemocionais. A curiosidade impulsiona a busca por novas soluções e o aprendizado contínuo, enquanto a resiliência é crucial para superar os desafios e erros comuns no processo de programação e depuração. A estrutura da aula, com exposição, prática e revisão, visa consolidar o conhecimento e incentivar a aplicação prática do que foi aprendido.
- **Exemplo Prático:** Durante a implementação de um método de autenticação, um estudante pode se deparar com erros de configuração ou lógica. A curiosidade o levará a pesquisar a causa do problema, e a resiliência o fará persistir na depuração até encontrar a solução, mesmo que leve tempo.

Referência do Slide: Slide 08 - Introdução à segurança em aplicações back-end

- **Definição:** A segurança em aplicações back-end é crucial para proteger dados sensíveis e garantir a confiabilidade do sistema. O slide enfatiza a importância de implementar boas práticas de segurança desde o início do desenvolvimento.
- **Aprofundamento/Complemento (se necessário):** A segurança não deve ser uma preocupação secundária ou um "patch" aplicado no final do projeto. Ela deve ser integrada em todas as etapas do ciclo de vida do desenvolvimento de software (SDLC), desde o design até a implantação e manutenção. Ataques cibernéticos são cada vez mais sofisticados, e uma única vulnerabilidade pode comprometer todo o sistema, resultando em perda de dados, danos à reputação e prejuízos financeiros.
- **Exemplo Prático:** Ao desenvolver uma API REST para um aplicativo bancário, é fundamental que todas as requisições sejam validadas e que os dados trafeguem de forma criptografada para evitar interceptações e acessos não autorizados. Ignorar a segurança nesse cenário seria catastrófico.

Referência do Slide: Slide 09 - Fundamentos de autenticação

- **Definição:** O slide aborda os fundamentos da autenticação, explicando o processo de verificação de identidade.
- **Aprofundamento/Complemento (se necessário):** Autenticação é o processo de confirmar a identidade de um usuário, dispositivo ou sistema. Isso geralmente envolve a apresentação de credenciais (como nome de usuário e senha) que são então verificadas contra um registro pré-existente. É a porta de entrada para a autorização, que define o que o usuário autenticado pode fazer.
- **Exemplo Prático:** Quando você digita seu PIN em um caixa eletrônico, o banco está autenticando sua identidade para garantir que você é o titular da conta e não uma pessoa não autorizada tentando acessá-la.

Referência do Slide: Slide 10 - Tipos de autenticação

- **Definição:** O slide apresenta diferentes tipos de autenticação, como autenticação baseada em senha, multifator e por token.
- **Aprofundamento/Complemento (se necessário):**
  - **Autenticação baseada em senha:** O método mais comum, onde o usuário fornece uma senha que é comparada com uma versão hash armazenada. A segurança depende da complexidade da senha e do hash utilizado.
  - **Autenticação multifator (MFA):** Requer duas ou mais credenciais independentes para verificar a identidade. Isso aumenta significativamente a segurança, pois mesmo que uma credencial seja comprometida, o atacante ainda precisaria de outra. Exemplos incluem senhas e códigos enviados por SMS ou aplicativos autenticadores.
  - **Autenticação por token:** Utiliza um token (geralmente uma string criptografada) emitido após a primeira autenticação bem-sucedida. O token é então usado para autorizar requisições subsequentes sem a necessidade de reautenticação. JWT (JSON Web Tokens) são um exemplo popular em APIs RESTful.
- **Exemplo Prático:**
  - **Baseada em senha:** O login em um site de rede social.
  - **MFA:** O acesso ao seu internet banking, que além da senha, pode pedir um token gerado por um aplicativo no celular.
  - **Por token:** Um aplicativo móvel que, após o primeiro login do usuário, armazena um token para mantê-lo logado, evitando que ele precise digitar a senha a cada vez que abre o aplicativo.

Referência do Slide: Slide 11 - Implementação de métodos de autenticação em Python

- **Definição:** O slide foca na implementação prática de métodos de autenticação utilizando a linguagem Python, apresentando exemplos de código.

- **Aprofundamento/Complemento (se necessário):** Python, com seus diversos frameworks (como Flask, Django) e bibliotecas (como `hashlib` para hashing, `PyJWT` para JWT), oferece ferramentas robustas para implementar diferentes estratégias de autenticação. É fundamental entender como utilizar essas ferramentas de forma segura, evitando vulnerabilidades comuns como injeção SQL, XSS (Cross-Site Scripting) e ataques de força bruta.

- **Exemplo Prático:** Um exemplo simples de autenticação em Python usando senha com hashing:

```python
import hashlib

def criar_hash_senha(senha):
    return hashlib.sha256(senha.encode()).hexdigest()

def verificar_senha(senha_digitada, hash_armazenado):
    return criar_hash_senha(senha_digitada) == hash_armazenado

# Simulação de um banco de dados de usuários
usuarios_bd = {
    "admin": criar_hash_senha("senha123"),
    "usuario1": criar_hash_senha("abc@123")
}

# Tentativa de login
usuario_login = "admin"
senha_login = "senha123"

if usuario_login in usuarios_bd and verificar_senha(senha_login, usuarios_bd[u
    print(f"Usuário '{usuario_login}' autenticado com sucesso!")
else:
    print("Usuário ou senha inválidos.")

# Tentativa de login com senha incorreta
usuario_login_errado = "usuario1"
senha_login_errada = "senha_incorreta"

if usuario_login_errado in usuarios_bd and verificar_senha(senha_login_errada,
    print(f"Usuário '{usuario_login_errado}' autenticado com sucesso!")
else:
    print(f"Falha na autenticação para o usuário '{usuario_login_errado}'.")
```

Referência do Slide: Slide 12 - Atividade prática: Implementando login e registro

- **Definição:** Este slide propõe uma atividade prática para os alunos, que consiste em implementar um sistema básico de login e registro, aplicando os conceitos de autenticação aprendidos.
- **Aprofundamento/Complemento (se necessário):** A atividade prática é essencial para solidificar o aprendizado. Ao construir um sistema real (mesmo que simplificado), os alunos enfrentam desafios práticos e consolidam a teoria. A implementação de login e registro envolve o armazenamento seguro de senhas (usando hashing e salt, por exemplo), a validação de entrada de dados e o manuseio de sessões ou tokens.
- **Exemplo Prático:** Os alunos podem ser solicitados a criar um script Python que:

1. Permita que um novo usuário se registre, armazenando seu nome de usuário e uma senha hashed.
2. Permita que um usuário existente faça login, verificando as credenciais.
3. Simule o uso de um token simples após o login bem-sucedido para autorizar acesso a uma "área restrita".

Semana 11 - Aula 2

Tópico Principal da Aula: Segurança em aplicações back-end

Subtítulo/Tema Específico: Autorização baseada em funções

Código da aula: [SIS]ANO2C2B2S11A2

Objetivos da Aula:

- Implementar autorização baseada em funções.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Slide 05 - Objetivos da aula

- **Definição:** O principal objetivo desta aula é capacitar os estudantes a implementar a autorização baseada em funções, um mecanismo crucial para controlar o acesso a recursos e funcionalidades em aplicações back-end.
- **Aprofundamento/Complemento (se necessário):** Enquanto a autenticação verifica "quem você é", a autorização determina "o que você pode fazer". Em sistemas complexos, diferentes usuários têm diferentes níveis de permissão. A autorização baseada em funções (Role-Based Access Control - RBAC) é um modelo de controle de acesso amplamente utilizado, onde as permissões são associadas a funções (roles), e os usuários são atribuídos a essas funções. Isso simplifica a gestão de permissões, especialmente em sistemas com muitos usuários e recursos.
- **Exemplo Prático:** Em um sistema de gerenciamento de conteúdo, um "Administrador" pode criar, editar e excluir todo o conteúdo, enquanto um "Editor" pode apenas criar e editar seu próprio conteúdo, e um "Leitor" só pode visualizar o conteúdo.

Referência do Slide: Slide 06 - Habilidades socioemocionais e atividades

- **Definição:** Este slide reitera as habilidades socioemocionais de curiosidade e resiliência na resolução de problemas, e descreve as atividades da aula: exposição teórica, atividade prática e a revisão final ("Então, ficamos assim...").
- **Aprofundamento/Complemento (se necessário):** A complexidade da autorização, especialmente em sistemas distribuídos ou microsserviços, pode apresentar desafios significativos. A curiosidade em explorar novas abordagens e a resiliência para depurar problemas de permissão são qualidades valiosas para o desenvolvedor. A estrutura da aula busca balancear a teoria com a prática para uma compreensão completa.
- **Exemplo Prático:** Um estudante pode encontrar dificuldades ao configurar as permissões para diferentes funções em um aplicativo web. Sua resiliência o ajudará a revisar a documentação, experimentar diferentes configurações e depurar o código até que as permissões funcionem conforme o esperado.

Referência do Slide: Slide 08 - Introdução à autorização

- **Definição:** O slide introduz o conceito de autorização, explicando que ela define o que um usuário autenticado tem permissão para fazer em um sistema.
- **Aprofundamento/Complemento (se necessário):** Após um usuário ser autenticado, o sistema precisa decidir quais recursos ele pode acessar e quais operações ele pode realizar. A autorização é o processo de conceder ou negar acesso com base em regras e políticas predefinidas. É uma camada de segurança vital que impede que usuários mal-intencionados ou não autorizados acessem informações ou funções para as quais não têm permissão, mesmo que tenham conseguido se autenticar.
- **Exemplo Prático:** Após fazer login em um painel administrativo, a autorização determina se você pode acessar a seção de "configurações de sistema" (se for um administrador) ou apenas a seção de "relatórios" (se for um analista).

Referência do Slide: Slide 09 - Modelos de autorização

- **Definição:** Este slide apresenta diferentes modelos de autorização, com foco especial na autorização baseada em funções (RBAC).
- **Aprofundamento/Complemento (se necessário):**
    - **Autorização Baseada em Funções (RBAC):** É o modelo mais comum, onde as permissões são agrupadas em "funções" (ex: Administrador, Gerente, Usuário Comum). Os usuários são então atribuídos a uma ou mais funções. Isso simplifica a gestão de permissões, pois você gerencia funções, não permissões individuais para cada usuário.

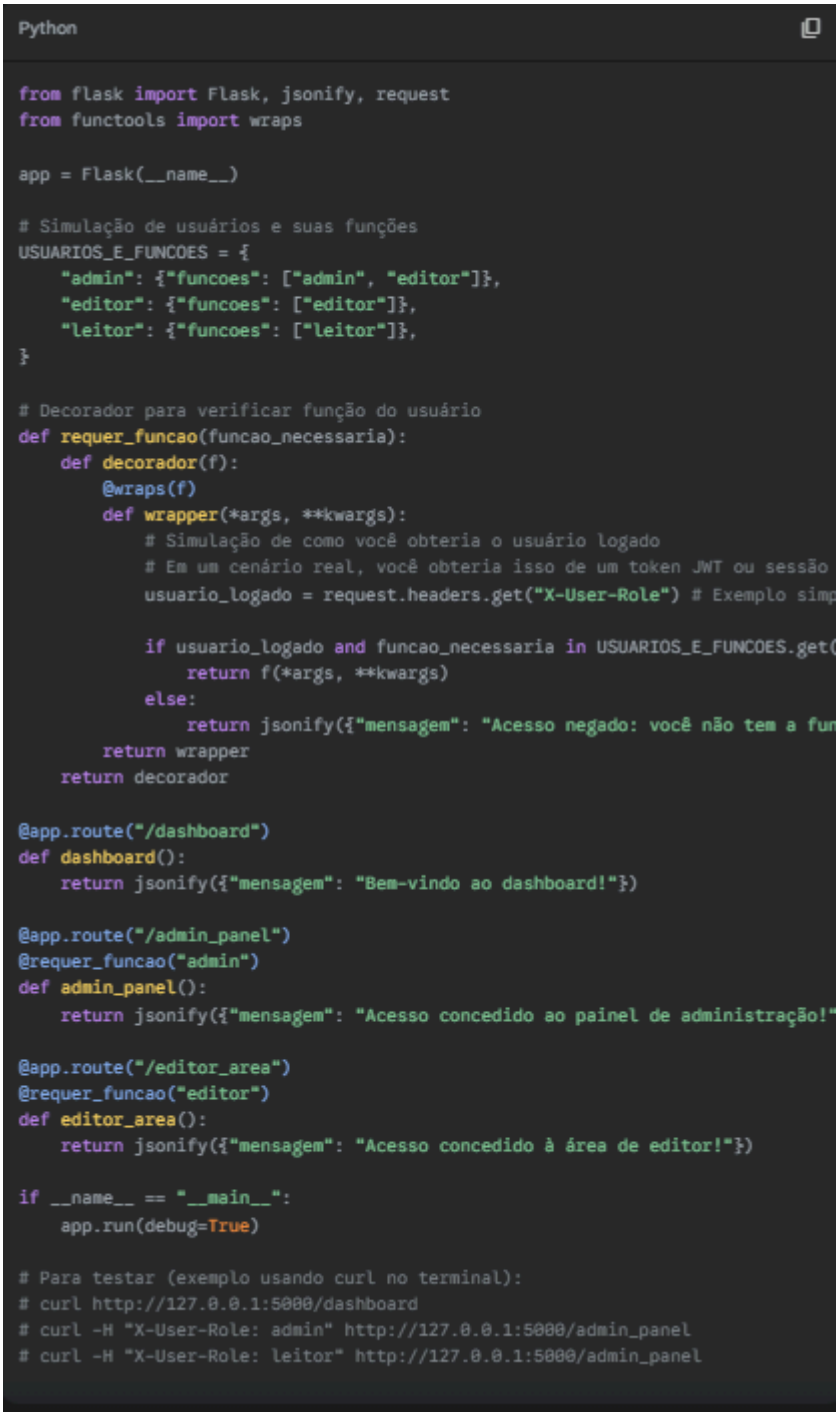
    - **Autorização Baseada em Atributos (ABAC):** Um modelo mais granular que concede acesso com base em atributos de usuário (ex: departamento, cargo), recursos (ex: tipo de dado, sensibilidade) e ambiente (ex: horário, localização). Mais flexível, mas também mais complexo de gerenciar.
    - **Autorização Baseada em Listas de Controle de Acesso (ACL):** As permissões são diretamente associadas a objetos e usuários. Cada objeto tem uma lista de usuários e suas permissões para aquele objeto. Pode ser difícil de escalar em sistemas grandes.
- **Exemplo Prático:** Em um sistema de e-mail corporativo:
    - **RBAC:** Um "Gerente de Equipe" tem permissão para ler e-mails de sua equipe, enquanto um "Funcionário" pode apenas ler e enviar seus próprios e-mails.
    - **ABAC:** Um funcionário do departamento financeiro pode acessar relatórios financeiros apenas durante o horário de trabalho e de um IP específico da empresa.

Referência do Slide: Slide 10 - Implementação de autorização em Python

- **Definição:** O slide aborda a implementação prática de autorização em aplicações Python, incluindo exemplos de código e a integração com frameworks populares.

- **Aprofundamento/Complemento (se necessário):** Em Python, frameworks web como Django e Flask possuem mecanismos de autorização integrados. Django, por exemplo, oferece um sistema de autenticação e permissões robusto com grupos e permissões por objeto. Flask permite a criação de decoradores personalizados para verificar funções antes de executar uma rota. O uso de bibliotecas como `Flask-Login` ou `Django-Auth` pode facilitar significativamente a implementação.

- **Exemplo Prático:** Implementação de um decorador em Flask para restringir o acesso a uma rota apenas para usuários com a função 'admin':



```python
from flask import Flask, jsonify, request
from functools import wraps

app = Flask(__name__)

# Simulação de usuários e suas funções
USUARIOS_E_FUNCOES = {
    "admin": {"funcoes": ["admin", "editor"]},
    "editor": {"funcoes": ["editor"]},
    "leitor": {"funcoes": ["leitor"]},
}

# Decorador para verificar função do usuário
def requer_funcao(funcao_necessaria):
    def decorador(f):
        @wraps(f)
        def wrapper(*args, **kwargs):
            # Simulação de como você obteria o usuário logado
            # Em um cenário real, você obteria isso de um token JWT ou sessão
            usuario_logado = request.headers.get("X-User-Role") # Exemplo simp

            if usuario_logado and funcao_necessaria in USUARIOS_E_FUNCOES.get(
                return f(*args, **kwargs)
            else:
                return jsonify({"mensagem": "Acesso negado: você não tem a fun
        return wrapper
    return decorador

@app.route("/dashboard")
def dashboard():
    return jsonify({"mensagem": "Bem-vindo ao dashboard!"})

@app.route("/admin_panel")
@requer_funcao("admin")
def admin_panel():
    return jsonify({"mensagem": "Acesso concedido ao painel de administração!"

@app.route("/editor_area")
@requer_funcao("editor")
def editor_area():
    return jsonify({"mensagem": "Acesso concedido à área de editor!"})

if __name__ == "__main__":
    app.run(debug=True)

# Para testar (exemplo usando curl no terminal):
# curl http://127.0.0.1:5000/dashboard
# curl -H "X-User-Role: admin" http://127.0.0.1:5000/admin_panel
# curl -H "X-User-Role: leitor" http://127.0.0.1:5000/admin_panel
```

Referência do Slide: Slide 11 - Práticas recomendadas para autorização

- **Definição:** Este slide apresenta as melhores práticas para implementar a autorização de forma segura e eficiente em aplicações back-end.
- **Aprofundamento/Complemento (se necessário):**
    - **Princípio do Privilégio Mínimo:** Conceder apenas as permissões necessárias para que um usuário ou sistema execute suas tarefas.

Evita que um ataque explorando uma vulnerabilidade tenha acesso total.
    - **Segregação de Funções:** Separar as responsabilidades e garantir que nenhuma função tenha excesso de privilégios. Por exemplo, a pessoa que aprova pagamentos não deve ser a mesma que os executa.
    - **Auditoria e Monitoramento:** Registrar eventos de acesso e tentativas de acesso não autorizado. Isso permite detectar e responder a incidentes de segurança rapidamente.
    - **Validação de Entrada:** Sempre validar e sanitizar todas as entradas do usuário para prevenir ataques como injeção.
    - **Gerenciamento Centralizado de Funções:** Utilizar um sistema centralizado para definir e gerenciar funções e permissões, facilitando a manutenção e a escalabilidade.
- **Exemplo Prático:** Em uma aplicação de RH, um funcionário do departamento de folha de pagamento não deve ter acesso aos dados sensíveis de saúde dos funcionários, mesmo que ambos sejam funcionários. A aplicação deve aplicar o princípio do privilégio mínimo e a segregação de funções.

Referência do Slide: Slide 12 - Atividade prática: Definindo funções e permissões

- **Definição:** A atividade prática proposta neste slide visa que os alunos apliquem o conceito de autorização baseada em funções, definindo diferentes funções e associando-as a permissões específicas em um cenário de aplicação.
- **Aprofundamento/Complemento (se necessário):** Esta atividade pode envolver a criação de um esquema de banco de dados para armazenar usuários, funções e as relações entre eles, além de desenvolver a lógica no código para verificar as permissões antes de permitir o acesso a determinadas funcionalidades.
- **Exemplo Prático:** Proponha a criação de um sistema de blog simples. Os alunos devem definir as seguintes funções:
    - **Administrador:** Pode criar, editar, excluir posts e gerenciar usuários.
    - **Autor:** Pode criar, editar e excluir seus próprios posts.
    - **Leitor:** Pode apenas visualizar posts. Eles devem implementar a lógica para que cada função só possa realizar as ações permitidas.

Tópico Principal da Aula: Segurança em aplicações back-end

Subtítulo/Tema Específico: Segurança em microsserviços

Código da aula: [SIS]ANO2C2B2S11A3

Objetivos da Aula:

- Implementar práticas de segurança em arquitetura de microsserviços.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Slide 05 - Objetivos da aula

- **Definição:** O objetivo desta aula é capacitar os estudantes a implementar práticas de segurança eficazes em arquiteturas baseadas em microsserviços.
- **Aprofundamento/Complemento (se necessário):** Microsserviços trazem flexibilidade e escalabilidade, mas também introduzem novos desafios de segurança em comparação com arquiteturas monolíticas. A superfície de ataque potencial aumenta, e a comunicação entre os serviços precisa ser protegida. Cada microsserviço, embora menor, pode ter suas próprias vulnerabilidades e pontos de entrada, exigindo uma abordagem de segurança holística e descentralizada.
- **Exemplo Prático:** Em um sistema de e-commerce dividido em microsserviços (e.g., um para gerenciamento de produtos, outro para processamento de pedidos, e outro para autenticação), a segurança precisa ser pensada em cada serviço e na comunicação entre eles.

Referência do Slide: Slide 06 - Habilidades socioemocionais e atividades

- **Definição:** O slide destaca as habilidades socioemocionais de curiosidade e resiliência, e as atividades da aula: exposição, atividade prática e a seção de revisão "Então, ficamos assim...".
- **Aprofundamento/Complemento (se necessário):** A segurança em microsserviços é um campo em constante evolução. A curiosidade para se manter atualizado sobre as últimas ameaças e melhores práticas é vital. A resiliência será testada ao depurar problemas de segurança em um ambiente distribuído, onde a causa raiz pode ser mais difícil de identificar.
- **Exemplo Prático:** Durante a configuração de um gateway de API para microsserviços, um estudante pode enfrentar problemas de roteamento e autenticação. A resiliência e a curiosidade o guiarão na leitura da

documentação e na experimentação até que a configuração de segurança esteja correta.

Referência do Slide: Slide 08 - Desafios de segurança em microsserviços

- **Definição:** Este slide aborda os desafios únicos de segurança que surgem ao trabalhar com arquiteturas de microsserviços, em comparação com aplicações monolíticas.
- **Aprofundamento/Complemento (se necessário):** Microsserviços introduzem novos pontos de vulnerabilidade:
    - **Comunicação entre serviços:** Cada chamada de serviço para serviço (Service-to-Service Communication) precisa ser segura, geralmente usando mTLS (mutual TLS) ou JWTs para autenticação e autorização interna.
    - **Gerenciamento de segredos:** Credenciais de banco de dados, chaves de API e outras informações sensíveis precisam ser gerenciadas de forma segura e distribuída.
    - **Visibilidade e Monitoramento:** Rastrear ataques em um ambiente distribuído é mais complexo, exigindo logging e monitoramento centralizados.
    - **Gestão de APIs:** Aumenta o número de APIs expostas, cada uma delas um potencial ponto de entrada para ataques.
    - **Orquestração e Descoberta de Serviços:** Plataformas como Kubernetes trazem complexidades de segurança adicionais na configuração e no controle de acesso.
- **Exemplo Prático:** Em uma arquitetura monolítica, o banco de dados é acessado por uma única aplicação. Em microsserviços, múltiplos serviços podem acessar o mesmo banco de dados (ou bancos de dados diferentes), exigindo que cada conexão seja autenticada e autorizada individualmente, aumentando a complexidade de gerenciamento de segredos.

Referência do Slide: Slide 09 - Autenticação e autorização em microsserviços

- **Definição:** O slide explica como a autenticação e a autorização são aplicadas em um ambiente de microsserviços, destacando a necessidade de mecanismos que funcionem em um ambiente distribuído.
- **Aprofundamento/Complemento (se necessário):**
    - **Gateway de API:** Geralmente, a autenticação inicial do usuário final ocorre em um Gateway de API (API Gateway), que valida as credenciais e, em seguida, propaga a identidade do usuário (por exemplo, via JWT) para os serviços downstream.
    - **Autenticação entre serviços (Service-to-Service):** Para a comunicação interna entre microsserviços, tokens JWT ou mTLS (mutual Transport Layer Security) são comumente usados para garantir que apenas serviços autorizados possam se comunicar.

    - **Identidade e Acesso Federados:** Em sistemas maiores, pode-se usar provedores de identidade externos (como OAuth 2.0 e OpenID Connect) para gerenciar a autenticação e autorização.
- **Exemplo Prático:** Quando um usuário faz login em um aplicativo móvel (front-end), a requisição vai para um API Gateway. O Gateway autentica o usuário (usando, por exemplo, um JWT). Após a autenticação, o Gateway anexa o token JWT às requisições para os microsserviços internos. Cada microsserviço, ao receber uma requisição, pode verificar o JWT para garantir que a requisição é legítima e que o usuário tem permissão para a ação solicitada.

Referência do Slide: Slide 10 - Gerenciamento de segredos

- **Definição:** O slide discute a importância e as técnicas para o gerenciamento seguro de segredos (chaves de API, credenciais de banco de dados, etc.) em ambientes de microsserviços.
- **Aprofundamento/Complemento (se necessário):** Em um ambiente distribuído, hardcoding de segredos no código-fonte é uma péssima prática. Ferramentas como HashiCorp Vault, AWS Secrets Manager, Azure Key Vault ou Kubernetes Secrets (com atenção à segurança) são utilizadas para armazenar, versionar e gerenciar de forma segura as credenciais que os microsserviços precisam para operar. A rotação de segredos e o princípio do privilégio mínimo também são cruciais.
- **Exemplo Prático:** Um microsserviço que interage com um banco de dados externo não deve ter as credenciais do banco de dados hardcoded no código. Em vez disso, ele deve buscar essas credenciais em um serviço de gerenciamento de segredos seguro no momento da inicialização ou quando necessário, garantindo que as credenciais nunca sejam expostas.

Referência do Slide: Slide 11 - Monitoramento e auditoria em microsserviços

- **Definição:** O slide enfatiza a necessidade de monitoramento e auditoria robustos para detectar e responder a incidentes de segurança em uma arquitetura de microsserviços.
- **Aprofundamento/Complemento (se necessário):** Em um ambiente com muitos serviços, o rastreamento de eventos de segurança é fundamental. Isso inclui:
    - **Logs Centralizados:** Agregação de logs de todos os microsserviços em um local central para facilitar a análise e a detecção de anomalias (ELK Stack - Elasticsearch, Logstash, Kibana; ou Splunk).
    - **Métricas de Segurança:** Monitorar tentativas de login falhas, acessos não autorizados, tráfego de rede incomum, etc.
    - **Traceability (Rastreabilidade):** Capacidade de rastrear uma requisição completa através de múltiplos microsserviços para entender o fluxo e identificar pontos de falha ou ataques.

    - **Alertas:** Configurar alertas para eventos de segurança críticos.
- **Exemplo Prático:** Um sistema de monitoramento detecta um aumento súbito e incomum nas tentativas de login falhas em um microsserviço de autenticação. Isso pode ser um indicativo de um ataque de força bruta, e um alerta é disparado para a equipe de segurança investigar.

Referência do Slide: Slide 12 - Atividade prática: Protegendo comunicação entre serviços

- **Definição:** A atividade prática desafia os alunos a implementar mecanismos de segurança na comunicação entre dois microsserviços simulados.
- **Aprofundamento/Complemento (se necessário):** Esta atividade pode envolver a configuração de mTLS entre serviços, a utilização de JWTs para autenticação de serviço a serviço, ou a implementação de um padrão de gateway de API para intermediar a comunicação.
- **Exemplo Prático:** Os alunos podem criar dois microsserviços simples (por exemplo, um serviço de "usuários" e um serviço de "pedidos"). O serviço de "pedidos" precisa consultar o serviço de "usuários" para obter informações. A tarefa seria implementar um mecanismo (como um JWT assinado e verificado) para garantir que apenas o serviço de "pedidos" autorizado possa acessar a API do serviço de "usuários".

Referência do Slide: Slide 13 - Então, ficamos assim...

- **Definição:** Este slide serve como resumo e fechamento da aula, reforçando a importância de abordar a segurança em microsserviços de forma proativa e abrangente.
- **Aprofundamento/Complemento (se necessário):** A segurança em microsserviços é um campo complexo que exige uma mudança de mentalidade em relação às arquiteturas monolíticas. É fundamental que os desenvolvedores entendam os novos vetores de ataque e as melhores práticas para mitigar riscos, garantindo a robustez e a resiliência de suas aplicações distribuídas.
- **Exemplo Prático:** Discussão em sala de aula sobre a importância de um "shift-left" na segurança, onde a segurança é pensada e implementada desde as fases iniciais do desenvolvimento, e não apenas como uma preocupação tardia.
