at_S15_A1_SL15_backend

Roteiro de atividade prática

Nome: _____ Turma: _____

Título da atividade: Situação real

Pontos principais do conteúdo:

1. Introdução aos Servidores Web

- **Função Principal:** Um servidor web é um software (como Apache, Nginx, IIS) que roda em um computador (servidor) e tem como principal objetivo receber requisições de clientes (geralmente navegadores web) através do protocolo HTTP/HTTPS e retornar respostas.
- **Processo Básico:** O processo fundamental é:
    1. O cliente (navegador) solicita um recurso (uma página HTML, uma imagem, dados).
    2. O servidor web recebe e processa essa requisição.
    3. Ele localiza ou gera o recurso solicitado.
    4. Envia o recurso de volta para o cliente como uma resposta HTTP, que o navegador então renderiza.
- **Tipos de Conteúdo:** Servidores web podem entregar **conteúdo estático** (arquivos que não mudam, como HTML, CSS, imagens) ou **conteúdo dinâmico** (gerado em tempo real por uma aplicação backend, como Node.js, PHP, Python, Java, que se comunica com o servidor web).

2. Criação de APIs Seguras

- **Autenticação e Autorização:** São os pilares da segurança de APIs.
    - **Autenticação:** Confirma a identidade do cliente que está fazendo a requisição. Métodos comuns incluem o uso de **Tokens JWT (JSON Web Tokens)** e o padrão **OAuth 2.0**.

    - **Autorização:** Define o que um cliente autenticado tem permissão para fazer (ex: um usuário comum pode ler dados, mas apenas um administrador pode excluí-los).
- **Criptografia:** É fundamental usar **HTTPS (TLS)** para criptografar toda a comunicação entre o cliente e a API. Isso impede que os dados (incluindo tokens e informações sensíveis) sejam interceptados e lidos por terceiros.
- **Validação de Dados (Input Validation):** Toda e qualquer informação recebida pela API deve ser rigorosamente validada. Isso previne uma série de ataques, como **SQL Injection** e **Cross-Site Scripting (XSS)**, garantindo que apenas dados no formato esperado sejam processados.
- **Rate Limiting e Throttling:** Implementar limites no número de requisições que um cliente pode fazer em um determinado período de tempo é crucial para proteger a API contra ataques de negação de serviço (DoS) e abuso por parte de bots ou usuários mal-intencionados.

3. Testes de Segurança em Aplicações

- **Análise de Vulnerabilidades (Scanning):** Utilização de ferramentas automatizadas para varrer a aplicação em busca de vulnerabilidades conhecidas, como configurações inseguras, componentes desatualizados ou falhas de segurança comuns (listadas no OWASP Top 10).
- **Teste de Penetração (Pentest):** Uma abordagem mais aprofundada onde um especialista (ou uma equipe) simula um ataque hacker real contra a aplicação. O objetivo é explorar ativamente as vulnerabilidades para entender o impacto real que uma falha de segurança poderia causar.
- **SAST (Static Application Security Testing):** Análise do código-fonte da aplicação em busca de falhas de segurança sem executar o código. É útil para encontrar problemas como *SQL Injection* ou senhas "hardcoded" diretamente no código-fonte, geralmente integrado ao processo de desenvolvimento (CI/CD).
- **DAST (Dynamic Application Security Testing):** Análise da aplicação enquanto ela está em execução. A ferramenta interage com a aplicação como um usuário (ou um atacante) faria, enviando diferentes tipos de dados para encontrar vulnerabilidades em tempo de execução.

Situação-problema

Durante o evento de lançamento do aplicativo de entrega, você recebe um alerta de que o servidor está sobrecarregado e o aplicativo está respondendo lentamente. Os usuários estão reclamando de problemas para acessar o serviço, e o evento está em pleno andamento. Como líder da equipe técnica, você precisa decidir rapidamente o que fazer.

Responda:

1. Em um momento de alta pressão, como você pode priorizar suas ações para resolver o problema de desempenho do servidor?(Mínimo de 15 linhas)