
Semana 15 - Aula 1

Tópico Principal da Aula: Servidores web, APIs e segurança das aplicações

Subtítulo/Tema Específico: Introdução aos servidores web

Código da aula: [SIS] ANO2C2B3S15A1

Objetivos da Aula:

- Compreender o funcionamento de servidores web e configurar um servidor básico.
- Desenvolver a curiosidade ao explorar como os servidores web operam.

Recursos Adicionais:

- Caderno para anotações;
- Computador com acesso à internet.

Exposição do Conteúdo:

Referência do Slide: 07 e 08 - Funcionamento de servidores

- **Definição:** **Um servidor web é um software ou hardware que armazena os arquivos de um site (como documentos HTML, imagens, folhas de estilo CSS e arquivos JavaScript) e os entrega ao navegador do usuário final. Sua função primária é receber uma solicitação HTTP (request) de um cliente (como um navegador), processar essa solicitação e enviar uma resposta (response) de volta. Esse ciclo de requisição-resposta é a base para o funcionamento de praticamente toda a internet como a conhecemos.**
- **Aprofundamento/Complemento:** Existem diferentes tipos de servidores que podem trabalhar em conjunto para entregar uma aplicação completa:
    - **Servidor HTTP:** Especializado em lidar com solicitações e respostas do protocolo HTTP. Exemplos populares são o Apache e o Nginx.
    - **Servidor de Aplicação:** Responsável por executar a lógica de negócios e gerar conteúdo dinâmico. Ele processa código do lado do servidor (como Java, Python, ou Node.js) para criar páginas personalizadas para o usuário.
    - **Servidor de Arquivos:** Focado em gerenciar o armazenamento e a distribuição de arquivos, como downloads, imagens ou vídeos.
- **Exemplo Prático:** Quando você digita www.google.com em seu navegador, o navegador envia uma solicitação HTTP para os servidores do Google. O servidor HTTP do Google recebe essa solicitação, a encaminha para um servidor de aplicação que prepara a página de busca, e então o servidor HTTP envia a página HTML resultante como uma resposta HTTP de volta para o seu navegador, que a renderiza na tela.
- **Vídeos Sugeridos:**
    - O que é um Servidor?
    - https://youtu.be/LLKYNgiyRXM?si=ypMBg-zxD89z3L4O
    -

Referência do Slide: 09 a 12 - Configuração Básica de Servidores

- **Definição:** A configuração de um servidor web envolve uma série de passos, desde a escolha do software até a implementação de medidas de segurança.
- **Aprofundamento/Complemento:**
    - **Passo 1: Escolha do Servidor:** A escolha do software depende da necessidade do projeto.
        - **Apache:** Um dos servidores mais antigos e populares, conhecido por sua flexibilidade, vasta documentação e compatibilidade com inúmeros sistemas. É uma ótima escolha para hospedagem compartilhada e projetos que requerem módulos específicos.
        - **Nginx:** Conhecido por sua alta performance, eficiência e capacidade de lidar com um grande número de conexões simultâneas com baixo uso de memória. É ideal para sites com alto tráfego e para atuar como um *reverse proxy* ou *load balancer*.
    - **Passo 2: Instalação:** Em sistemas baseados em Linux (como Ubuntu/Debian), a instalação é feita através de comandos simples no terminal.
        - **Para Apache:** `sudo apt update && sudo apt install apache2`
        - **Para Nginx:** `sudo apt update && sudo apt install nginx`
    - **Passo 3: Configuração de Portas (Firewall):** É crucial permitir o tráfego nas portas padrão da web. O `ufw` (Uncomplicated Firewall) é uma ferramenta comum para isso.
        - **Porta 80 (HTTP):** Usada para tráfego não criptografado. Comando: `sudo ufw allow 80`
        - **Porta 443 (HTTPS):** Usada para tráfego seguro e criptografado. Comando: `sudo ufw allow 443`
    - **Passo 4: Segurança (Certificados SSL):** Certificados SSL são essenciais para criptografar a comunicação entre o cliente e o servidor (ativando o HTTPS). O Let's Encrypt oferece certificados gratuitos, e o `certbot` é uma ferramenta que automatiza a instalação.
        - Comando para Nginx: `sudo certbot --nginx`
- **Exemplo Prático:** Após instalar o Nginx (`sudo apt install nginx`) e permitir o tráfego no firewall (`sudo ufw allow 'Nginx Full'`), você pode acessar o endereço IP do seu servidor em um navegador e verá a página de boas-vindas padrão do Nginx, confirmando que a instalação foi bem-sucedida.
- **Vídeos Sugeridos:**
    - Instalando Servidor Web Apache no Ubuntu (PETERLINUX)
    - Let's Encrypt: HTTPS de graça (Código Fonte TV)

Semana 15 - Aula 2

Tópico Principal da Aula: Servidores web, APIs e segurança das aplicações

Subtítulo/Tema Específico: Criação de APIs seguras

Código da aula: [SIS] ANO2C2B3S15A2

Objetivos da Aula:

- Implementar técnicas de segurança em APIs RESTful.
- Compreender a importância da autenticação, autorização e criptografia para proteger dados.

Recursos Adicionais:

- Caderno para anotações;
- Computador com acesso à internet.

Exposição do Conteúdo:

Referência do Slide: 10 (Guia do Professor) - Autenticação e Autorização

- **Definição:**
  - **Autenticação:** É o processo de verificar quem um usuário ou sistema é. Essencialmente, responde à pergunta "Quem é você?". Em APIs, isso geralmente é feito através de chaves de API, tokens (como JWT - JSON Web Tokens) ou credenciais de login.
  - **Autorização:** Após a autenticação, a autorização é o processo que determina o que um usuário autenticado tem permissão para fazer. Ela responde à pergunta "O que você pode fazer?".
- **Aprofundamento/Complemento:** A falta de uma distinção clara entre esses dois conceitos é uma falha de segurança comum. Uma API precisa primeiro confirmar a identidade do solicitante (autenticação) e, em seguida, verificar se essa identidade tem os privilégios necessários para acessar o recurso ou executar a ação solicitada (autorização). Em uma API de pagamentos, por exemplo, a autenticação é vital para garantir que apenas usuários legítimos iniciem transações, prevenindo fraudes.
- **Exemplo Prático:** Em uma plataforma de e-commerce, um cliente (usuário comum) e um administrador fazem login com suas credenciais (autenticação). O cliente é **autorizado** a ver e gerenciar apenas seus próprios pedidos. Já o administrador, com um nível de permissão diferente, está **autorizado** a acessar um painel com relatórios financeiros globais e gerenciar todos os pedidos da plataforma.
- **Vídeos Sugeridos:**
  - Autenticação vs. Autorização (Código Fonte TV)
  - O que é JWT (JSON Web Token)? (Attekita Dev)

Referência do Slide: 11 (Guia do Professor) - Criptografia (HTTPS e Dados em Repouso)

- **Definição:**
  - **HTTPS (Criptografia em Trânsito):** O uso de HTTPS é fundamental para proteger os dados enquanto são transferidos entre o cliente e o servidor. Ele utiliza criptografia (via SSL/TLS) para impedir que invasores, como em ataques *man-in-the-middle*, possam interceptar e ler informações sensíveis como senhas, dados de cartão de crédito ou informações pessoais.
  - **Criptografia de Dados em Repouso:** Refere-se à proteção dos dados enquanto estão armazenados em um banco de dados ou em arquivos no servidor. Mesmo que um invasor consiga acesso físico ao servidor ou uma cópia do banco de dados, os dados criptografados permanecerão ilegíveis sem a chave de descriptografia.
- **Aprofundamento/Complemento:** Para senhas, nunca se deve armazená-las como texto simples. É essencial usar algoritmos de

hashing fortes e com "sal" (um valor aleatório adicionado à senha antes do hash), como o **bcrypt** ou **Argon2**. Esses algoritmos são projetados para serem lentos, o que torna ataques de força bruta (tentar adivinhar senhas em massa) muito mais difíceis e demorados.

- **Exemplo Prático:** Ao fazer uma compra online, o HTTPS garante que os dados do seu cartão de crédito sejam embaralhados e enviados de forma segura do seu navegador para o servidor da loja. Uma vez no servidor, a criptografia de dados em repouso garante que o número do seu cartão, se precisar ser armazenado, esteja protegido no banco de dados da empresa. A combinação das duas práticas oferece uma proteção robusta e completa.
- **Vídeos Sugeridos:**
    - Como funciona a Criptografia de ponta-a-ponta? (Canaltech)
    - NUNCA armazene senhas em texto puro! (Filipe Deschamps)

Semana 15 - Aula 3

Tópico Principal da Aula: Servidores web, APIs e segurança das aplicações

Subtítulo/Tema Específico: Testes de segurança em aplicações

Código da aula: [SIS] ANO2C2B3S15A3

Objetivos da Aula:

- Realizar testes de segurança para identificar e corrigir vulnerabilidades.
- Compreender como testes de penetração e ferramentas automatizadas são usados para mitigar riscos.

Recursos Adicionais:

- Caderno para anotações;
- Computador com acesso à internet.

Exposição do Conteúdo:

Referência do Slide: 10 (Guia do Professor) - Testes de Penetração (Pen Tests)

- **Definição: Testes de penetração, também conhecidos como *Pen Tests* ou hacking ético, são simulações autorizadas de ataques cibernéticos contra um sistema, rede ou aplicação web. O objetivo é identificar e explorar vulnerabilidades de segurança de forma controlada, antes que um invasor mal-intencionado o faça.**
- **Aprofundamento/Complemento:** Realizar Pen Tests antes do lançamento de uma aplicação é uma etapa crucial do ciclo de vida de desenvolvimento de software. Eles ajudam a garantir a segurança dos dados dos usuários, proteger a aplicação contra acessos não autorizados, e prevenir vazamentos de informações que poderiam causar danos financeiros e de reputação à empresa. Para uma corretora de seguros, por exemplo, um Pen Test é vital para garantir que dados sensíveis de clientes permaneçam seguros.
- **Exemplo Prático:** Uma equipe de segurança é contratada para realizar um Pen Test em um novo aplicativo de banco digital. Eles tentam ativamente explorar falhas, como tentar acessar a conta de outro cliente alterando parâmetros na URL, injetar código malicioso em campos de formulário ou sobrecarregar o servidor para tirá-lo

do ar. Ao final, eles entregam um relatório detalhado com todas as falhas encontradas para que a equipe de desenvolvimento possa corrigi-las.

- **Vídeos Sugeridos:** **A Forma Correta de Criptografar Senhas: O Guia que Todo Dev Precisa Ver!**
- **https://youtu.be/sC-rrOdwsIs?si=UttmZtNp_jLB3Rqs**

Referência do Slide: 10 e 11 (Guia do Professor) - Ferramentas e Tipos de Vulnerabilidades

- **Definição:** Ferramentas de teste de segurança automatizam o processo de encontrar falhas comuns. O
  OWASP ZAP (Zed Attack Proxy) é uma ferramenta de código aberto popular que atua como um "proxy de ataque", interceptando o tráfego entre o navegador e a aplicação para analisar e modificar as solicitações em busca de vulnerabilidades.
- **Aprofundamento/Complemento:** Durante os testes, seja manual ou com ferramentas, é fundamental verificar um conjunto de vulnerabilidades críticas, muitas das quais estão listadas no OWASP Top 10:
  - **Injeção de SQL (SQL Injection):** Permite que um invasor execute comandos SQL maliciosos no banco de dados da aplicação, podendo roubar, alterar ou excluir dados.
  - **Cross-Site Scripting (XSS):** Ocorre quando um script malicioso é injetado em uma página web e executado no navegador de um usuário, permitindo o roubo de cookies de sessão ou outras informações.
  - **Exposição de Dados Sensíveis:** Ocorre quando informações como senhas, números de cartão de crédito ou dados pessoais são armazenadas ou transmitidas sem a devida criptografia.
  - **Falhas de Autenticação e Autorização:** Permitem que invasores assumam a identidade de outros usuários ou acessem recursos restritos para os quais não têm permissão.
  - **Configurações de Segurança Inadequadas:** Inclui falhas como manter senhas padrão, habilitar serviços desnecessários ou exibir mensagens de erro detalhadas que expõem informações sobre o sistema.
- **Exemplo Prático:** Usando o OWASP ZAP, um analista de segurança configura seu navegador para passar todo o tráfego pela ferramenta. Ao navegar na aplicação, o ZAP escaneia automaticamente cada página e endpoint da API, simulando ataques como injeções de SQL nos campos de login. Se uma vulnerabilidade for encontrada, a ferramenta a reporta com detalhes sobre como explorá-la e corrigi-la.
