**Segurança de Aplicações Web – Resumo e Conceitos Fundamentais**

**O que é uma aplicação web?**

Uma **aplicação web** é um programa acessado diretamente por navegadores modernos (Chrome, Firefox, Safari) sem necessidade de instalação local. Exemplos comuns incluem:

* **Serviços de e-mail**: Tutanota, Protonmail, Outlook, Gmail
* **Suítes de escritório online**: Microsoft Office 365, Google Drive, Zoho Office
* **E-commerce**: Amazon, AliExpress, Etsy
* **Outros**: bancos online, previsão do tempo, redes sociais

Essas aplicações rodam em **servidores remotos**, que interagem com **bancos de dados** para recuperar e armazenar informações como produtos, clientes e vendas.

**Exemplo de funcionamento básico (site de compras):**

1. Usuário pesquisa por um item.
2. O navegador envia a pesquisa ao servidor.
3. O servidor consulta o banco de dados de produtos.
4. Os resultados são formatados e enviados de volta ao usuário como página web.

**Riscos de Segurança em Aplicações Web**

Aplicações web são alvos frequentes de ataques, principalmente porque armazenam informações sensíveis. Algumas empresas, como Google, Microsoft e Facebook, possuem programas de **recompensa por bugs** (bug bounty), incentivando a descoberta ética de vulnerabilidades.

**Exemplo típico de fluxo de um usuário:**

1. Fazer login
2. Pesquisar produtos
3. Adicionar ao carrinho
4. Informar endereço
5. Informar pagamento

Cada uma dessas etapas pode ser explorada por invasores, como veremos abaixo.

**Principais Vulnerabilidades em Aplicações Web**

**1. Falhas de Identificação e Autenticação**

* **Identificação**: reconhecer um usuário único.
* **Autenticação**: verificar se o usuário é realmente quem diz ser.

**Vulnerabilidades comuns:**

* Permitir ataques de força bruta (tentativas automatizadas de senhas).
* Aceitar senhas fracas.
* Armazenar senhas em **texto simples** (sem criptografia).

**Exemplo:** Tabela de banco de dados com senhas visíveis.

**2. Controle de Acesso Quebrado (Broken Access Control)**

Garante que usuários só acessem o que têm permissão.

**Erros comuns:**

* Usuários com mais permissões que o necessário.
* Conseguir visualizar ou editar dados de outros usuários.
* Acessar páginas restritas sem estar autenticado.

**Exemplo real:**

* Um cliente acessando a página user?id=16 e depois trocando para id=17, acessando os dados de outro usuário.

**3. Injeção (Injection)**

Quando o sistema aceita **entradas maliciosas** do usuário, que são executadas como comandos.

**Causa:** falta de validação e sanitização de entrada.

**Exemplo:** Digitar códigos em campos de pesquisa para forçar o sistema a revelar dados indevidos.

**4. Falhas Criptográficas**

Referem-se a erros no uso de criptografia, que é essencial para proteger dados.

**Vulnerabilidades comuns:**

* Enviar dados sensíveis em **texto claro** (ex: usando HTTP ao invés de HTTPS).
* Utilizar **algoritmos fracos**, como cifras simples de substituição.
* Usar **chaves fracas ou padrão** (ex: "1234").

**Exemplo:** Um número de cartão de crédito enviado pela internet sem criptografia.

**Vulnerabilidade: IDOR (Insecure Direct Object References)**

**O que é:**  
Acontece quando o sistema confia demais na entrada fornecida pelo usuário sem validar se ele tem permissão para acessar um recurso.

**Exemplo prático:**

* URL: https://store.exemplo.com/products?id=52  
  Um invasor pode tentar acessar id=51, id=53, etc., acessando produtos ou contas não autorizadas.

Essa falha é uma forma de **Controle de Acesso Quebrado** e pode ser explorada para visualizar ou alterar dados de terceiros.

**Estudo de Caso: Sabotagem via IDOR**

* Um sistema de **gerenciamento de inventário** teve sabotagem via IDOR.
* Atacante alterou entregas de pneus incorretos para linhas de montagem.
* A tarefa do usuário é **reverter a sabotagem** e corrigir os dados.

**Conclusão**

Aplicações web, apesar de convenientes, exigem atenção rigorosa à segurança. As falhas descritas, como controle de acesso quebrado, autenticação fraca e injeções, podem comprometer seriamente sistemas e dados de usuários. Entender e identificar essas vulnerabilidades é o primeiro passo para construir aplicações mais seguras.