Exemplos práticos e fáceis de entender para cada categoria do método STRIDE

🔷 S – Spoofing (Falsificação de Identidade)

📌 Exemplo: Um site de login falso que imita o original para roubar credenciais.
💡 Explicação: Imagine que um hacker cria uma página idêntica ao seu sistema de login. Quando um usuário digita seu e-mail e senha, esses dados são enviados para o hacker em vez do sistema real.

✅ Prevenção:

- Usar autenticação multifator (MFA).
- Verificar certificados SSL/TLS no navegador.
- Implementar tokens de sessão seguros.

🔷 T – Tampering (Adulteração de Dados)

📌 Exemplo: Um usuário mal-intencionado altera o preço de um produto em um e-commerce antes de finalizar a compra.
💡 Explicação: No carrinho de compras, ao inspecionar os elementos da página (via DevTools do navegador), o usuário pode modificar o preço e pagar um valor muito menor.

✅ Prevenção:

- Validar os preços no servidor, não apenas no frontend.
- Usar assinaturas digitais ou hashes para validar dados.
- Evitar confiar em valores enviados pelo usuário sem verificação.

🔷 R – Repudiation (Repúdio)

📌 Exemplo: Um usuário faz uma transferência de dinheiro e depois alega que não realizou a operação.
💡 Explicação: Sem registros adequados, não há como provar se a transação foi realmente feita por ele.

✅ Prevenção:

- Implementar logs de auditoria com timestamps.
- Registrar todas as ações sensíveis dos usuários (com hashes ou assinaturas digitais).
- Utilizar autenticação forte para operações críticas.

🔷 I – Information Disclosure (Exposição de Informações)

📌 Exemplo: Um erro no site exibe credenciais do banco de dados ao usuário.
💡 Explicação: Se um erro de conexão com o banco de dados ocorre e a mensagem de erro completa é exibida, um invasor pode obter detalhes como nome do banco, usuário e senha.

✅ Prevenção:

- Desativar mensagens de erro detalhadas em produção.
- Nunca expor credenciais no código-fonte público.
- Criptografar informações sensíveis.

🔷 D – Denial of Service (Negação de Serviço)

📌 Exemplo: Um invasor sobrecarrega um site enviando milhões de requisições simultaneamente.
💡 Explicação: Se um site recebe um volume muito alto de acessos artificiais, ele pode ficar lento ou sair do ar, impedindo usuários legítimos de acessá-lo.

✅ Prevenção:

- Usar serviços de proteção contra DDoS (Cloudflare, AWS Shield).
- Implementar limites de requisições (Rate Limiting).
- Monitorar tráfego anormal e bloquear IPs suspeitos.

🔷 E – Elevation of Privilege (Elevação de Privilégio)

📌 Exemplo: Um usuário comum altera sua permissão para "administrador" manipulando requisições.
💡 Explicação: Se um sistema verifica permissões apenas no frontend, um atacante pode modificar a solicitação enviada ao servidor e obter acesso a funcionalidades restritas.

✅ Prevenção:

- Verificar permissões no servidor, não apenas no frontend.
- Usar princípios de menor privilégio (cada usuário só deve acessar o que precisa).
- Auditar permissões regularmente.

1-token_seguro.html

🔷 🛡️ Segurança Implementada

✅ Regeneração de sessão: session_regenerate_id(true); impede sequestro de sessão.
✅ Tokens seguros: bin2hex(random_bytes(32)) cria tokens imprevisíveis.
✅ Verificação no banco: O sistema compara o token da sessão com o armazenado no banco.
✅ Destruição segura: Ao fazer logout, o token é removido do banco e a sessão é encerrada.

Esse é um modelo simples e seguro para lidar com tokens de sessão! 🚀

🔷 **Como Funciona**

1. O usuário insere as credenciais e clica em **"Entrar"**.
    - Se forem válidas (admin@example.com e 123456), um **token seguro** é gerado e salvo.
    - A animação de **carregamento** simula o processamento do login.
    - O status muda para **"Login bem-sucedido!"**, e o token é exibido.
2. Ao clicar em **"Verificar Sessão"**, o sistema verifica se há um token salvo.
    - Se existir, o status mostra **"Sessão ativa!"**.
    - Se não existir, mostra **"Nenhuma sessão encontrada."**
3. No **"Logout"**, o token é removido e a animação exibe a finalização da sessão.

2-adulteracao.html

O problema: O usuário pode manipular o preço do produto no frontend antes de enviá-lo ao servidor.
A solução: O backend valida o preço real, usando um hash para impedir manipulação.

🛠 Como funciona o código?

✅ **O usuário altera o preço manualmente no frontend.**
✅ **O sistema envia um hash seguro para validar o preço no backend.**
✅ **Se o hash não corresponder, o sistema detecta adulteração.**

No exemplo atual, se você abrir o Inspecionar Elemento (F12) → Console e rodar:

`document.getElementById("preco").value = 10;`

O preço será alterado no frontend sem que o usuário perceba. Isso acontece porque o código confia no valor inserido no campo de entrada antes de enviar os dados.

O problema: Um usuário pode realizar ações críticas e negar que as fez.
A solução: O sistema registra todas as ações com timestamp, ID do usuário e assinatura digital.

🛠 Como funciona o código?

✅ Cada ação do usuário é registrada nos logs.
✅ O log inclui um timestamp e um hash único.
✅ Se houver manipulação, o hash não será mais válido.

3-repudio.html

O problema: Um usuário pode realizar ações críticas e negar que as fez.
A solução: O sistema registra todas as ações com timestamp, ID do usuário e assinatura digital.

🛠 Como funciona o código?

✅ Cada ação do usuário é registrada nos logs.
✅ O log inclui um timestamp e um hash único.
✅ Se houver manipulação, o hash não será mais válido.

4-exposicao_informacoes.html

O problema: Um erro no site exibe informações sensíveis como credenciais do banco.
A solução:
✅ Ocultar mensagens detalhadas de erro em produção.
✅ Nunca expor credenciais no código-fonte.
✅ Utilizar logs seguros sem informações sensíveis.

5-negacao_servico.html

Clique em "Acessar Site" → O site responde normalmente.
Clique várias vezes seguidas → Depois de 5 tentativas, o IP é bloqueado.
Clique em "Simular Ataque 🚨" → Várias requisições são feitas ao mesmo tempo e o IP é bloqueado!

🛡️ **Como evitar ataques DDoS?**

✅ **Utilizar Rate Limiting (limite de requisições) → Como no exemplo, bloquear IPs que fazem muitas requisições.**

✅ **Monitorar tráfego anormal → Se houver picos repentinos, bloquear IPs suspeitos.**

✅ **Usar serviços de proteção → Como Cloudflare, AWS Shield, Akamai.**

✅ **Configurar firewall → Pode bloquear IPs maliciosos automaticamente.**

6-elevacao_privilegios.html

O problema: Um usuário comum pode modificar o nível de permissão via inspeção de elementos no navegador.
A solução:
✅ **Verificar permissões no servidor, não apenas no frontend.**
✅ **Proteger endpoints críticos contra alterações maliciosas.**
✅ **Usar tokens de autenticação seguros (JWT, sessões) para garantir privilégios corretos.**

Tente inspecionar o código e alterar nivelAcesso = "admin" no console.

Como alterar nivelAcesso no console do navegador:

Abra a página no navegador (Chrome, Edge, Firefox, etc.).

Acesse o console do desenvolvedor:

- **Windows/Linux: Pressione F12 ou Ctrl + Shift + I e vá até a aba "Console".**
- **Mac: Pressione Cmd + Option + I.**

Digite o seguinte comando no console:

```
nivelAcesso = "admin";
```

Isso altera a variável nivelAcesso no frontend, simulando um ataque.

Clique no botão "Tentar Acessar Área Administrativa".
O usuário sem permissão agora acessa a área restrita.

🛡️ **Como evitar a Elevação de Privilégio?**

✅ **Verificar privilégios no servidor, não apenas no frontend!**
➡️ **Exemplo seguro em PHP:**

```
session_start();

if ($_SESSION['role'] !== 'admin') {
```

```
    die("Acesso negado! 🚫");
}
```

✅ **Usar tokens seguros para autenticação (JWT, sessões).**
✅ **Não confiar em valores enviados pelo usuário!**

🔷 **O que aprendemos?**

🔒 **Frontend NÃO é lugar para validação de segurança!**
🔒 **Verifique privilégios no backend antes de conceder acesso!**
🔒 **Use sessões seguras para autenticação!**