Checklist de práticas de segurança em software

1. Planejamento e Requisitos

- [ ] Definir *requisitos de segurança e privacidade* desde o início.
- [ ] Realizar análise de riscos e ameaças (Threat Modeling).
- [ ] Estabelecer políticas de segurança e conformidade (ex: LGPD, ISO 27001).
- [ ] Incluir *critérios de segurança* nos contratos com terceiros.

2. Design e Arquitetura

- [ ] Aplicar o princípio de "*segurança por design*".
- [ ] Minimizar a superfície de ataque (ex: *desabilitar portas e serviços desnecessários*).
- [ ] Escolher *algoritmos criptográficos seguros* e atualizados.
- [ ] *Documentar fluxos de dados* sensíveis e pontos críticos.

3. Implementação

- [ ] Seguir práticas de *codificação segura* (ex: OWASP Top 10).
- [ ] *Validar e sanitizar* todas as entradas de usuários.
- [ ] Evitar hardcoded secrets (usar cofres de segredos).
- [ ] Utilizar ferramentas de análise estática e linters com foco em segurança.

4.Testes e Validação

- [ ] Realizar *testes de segurança automatizados* (SAST, DAST, SCA).
- [ ] Executar testes de penetração e fuzzing em componentes críticos.
- [ ] Testar *autenticação, autorização e controle* de sessões.
- [ ] *Corrigir vulnerabilidades* antes da liberação.

5. Deploy e Entrega

- [ ] Automatizar o pipeline CI/CD com validações de segurança.
- [ ] *Assinar artefatos* e verificar integridade de dependências.
- [ ] Configurar ambientes com o *menor privilégio* necessário.
- [ ] *Documentar* procedimentos de rollback seguros.

6. Operação e Monitoramento

- [ ] Monitorar logs de segurança e alertas em *tempo real*.
- [ ] Implementar detecção de anomalias e respostas a incidentes.

- [ ] *Atualizar* e aplicar patches regularmente.
- [ ] Realizar *auditorias e revisões* de segurança periódicas.

7. Melhoria Contínua

- [ ] Promover *treinamentos regulares* sobre segurança para a equipe.
- [ ] *Revisar e atualizar* políticas de segurança conforme necessário.
- [ ] Incorporar *feedbacks de incidentes* para fortalecer processos.
- [ ] *Manter-se atualizado* com as últimas ameaças e vulnerabilidades.

O que foi aprendido

Segurança como parte integral do desenvolvimento: A segurança não é um complemento, mas uma *parte essencial* de cada fase do desenvolvimento de software.

Práticas fundamentais de segurança: Incluem validação de entradas, uso adequado de criptografia, controle de acessos e monitoramento contínuo.

Ferramentas e metodologias recomendadas: Utilização de ferramentas como SAST, DAST, SCA, além de práticas como modelagem de ameaças e análise de riscos.

Por que isso é importante

Redução de riscos: *Identificar e corrigir vulnerabilidades* precocemente evita explorações maliciosas e danos à reputação da empresa.

Conformidade regulatória: *Atender a normas e regulamentações* como LGPD, ISO 27001, entre outras.

Economia de recursos: *Correções no início do desenvolvimento* são menos custosas do que após o software estar em produção.

Analogia com o mundo real

Imagine construir uma casa:
Planejamento sem segurança: É como projetar uma casa *sem considerar* fechaduras ou sistemas de alarme.

Construção negligente: Usar materiais de *baixa qualidade* ou não seguir normas de segurança estrutural.

Manutenção inexistente: *Ignorar rachaduras ou infiltrações* que surgem com o tempo.

> Assim como uma casa *requer segurança desde o projeto até a manutenção*, o software também precisa ser protegido em todas as suas fases.

Erros comuns e como evitar

1. **Ignorar a segurança no planejamento**: *Não considerar requisitos de segurança* desde o início.
   Como evitar: *Incluir especialistas em segurança* nas fases iniciais e realizar análise de riscos.
2. **Validação inadequada de entradas**: *Não verificar ou sanitizar dados* fornecidos por usuários.
   Como evitar: *Implementar validações rigorosas* e utilizar bibliotecas confiáveis para sanitização.
3. **Armazenamento de credenciais em código**: Incluir *senhas ou chaves diretamente* no código-fonte.
   Como evitar: *Utilizar gerenciadores de segredos* e variáveis de ambiente.
4. **Falta de testes de segurança**: *Não realizar testes específicos* para identificar vulnerabilidades.
   Como evitar: *Integrar testes de segurança automatizados e manuais* no processo de desenvolvimento.
5. **Atualizações negligenciadas**: *Não aplicar patches ou atualizações de segurança*.
   Como evitar: Estabelecer processos para *monitoramento e aplicação regular de atualizações*.

Recomendações de leitura

1. OWASP - Secure Coding Practices
2. NIST - Secure Software Development Framework (SSDF)
3. Microsoft - Security Development Lifecycle (SDL)
4. Red Hat - Security in the software development lifecycle
5. Snyk - Secure Software Development Lifecycle (SSDLC)
6. Langate - Common Security Mistakes in Software Development
7. Codecademy - All about the Secure Software Development Lifecycle (SSDLC)

Simulação de fluxo seguro em aplicativo bancário

Esse fluxograma foi pensado para mostrar, de forma clara e prática, como um aplicativo bancário pode (e deve) proteger o usuário em cada etapa do processo de acesso e transação.

> Usei cores diferentes para facilitar a visualização e associar cada etapa a um status de segurança, seguindo boas práticas que aprendi estudando Engenharia de Software Seguro.

1. Entrada do Usuário (Azul):
Tudo começa quando o usuário tenta acessar o app. O azul aqui representa confiança, aquele momento em que o usuário chega e espera ser bem recebido, mas com segurança.

2. Verificação de Privilégios (Verde):
Logo de cara, o sistema verifica se o usuário tem permissão para fazer o que está tentando. Verde porque, se está tudo certo, é sinal de “pode passar”. Se não tiver permissão, o caminho já leva direto para a notificação de acesso negado.

3. Notificação de Acesso Negado (Vermelho):
Se o usuário não tiver privilégio suficiente, o app bloqueia o acesso e já manda um aviso. Vermelho porque é alerta máximo: “parou por aqui, acesso negado!”.

4. Monitoramento de Atividade (cinza):
Se o usuário passou na verificação, o sistema começa a monitorar tudo em tempo real. Cinza porque é neutro, fica ali de olho, sem interferir – só observando o que acontece.

5. Atividade Suspeita (Laranja):
Se algo estranho acontece (tipo uma tentativa de acesso de outro país, ou um valor fora do comum), o sistema acende o alerta laranja: “atenção, pode ter coisa errada aqui”.

6. Autenticação Forte em Transações Suspeitas (Roxo):
Se a atividade for mesmo suspeita, entra a autenticação reforçada – biometria, reconhecimento facial ou outro método forte. Roxo porque é uma etapa especial, de segurança extra.

7. Processamento da Transação (Azul Turquesa):
Se não for nada suspeito, a transação segue normalmente. O azul turquesa mostra que está tudo fluindo, processo em andamento.

8. Notificação ao Usuário (Amarelo):
Independente do que aconteceu, o usuário sempre é avisado. Amarelo porque chama a atenção, é aquele “olha, aconteceu algo importante, fique ligado”.

9. Fim do Processo (Verde Escuro):
Depois de tudo, o processo é finalizado com sucesso. Verde escuro porque representa missão cumprida: seguro, confiável e sem surpresas.

> `Esse fluxograma mostra que segurança não é só uma etapa, mas um ciclo contínuo de monitoramento, prevenção e resposta. O usuário é protegido em cada passo, mas também é informado e envolvido, porque segurança de verdade é feita em conjunto de sistema e pessoa, cada um fazendo sua parte.`

Recomendações de leitura

1. O que é Internet Banking? Saiba como funciona e para que serve!
2. Dicas Essenciais de Segurança ao Usar Aplicativos de Banco no Celular
3. O que é segurança móvel? Benefícios, ameaças e práticas recomendadas
4. Por que a Segurança de Dispositivos Móveis é tão importante?