✱ Boas Práticas de automação

↳ AÇÃO SUT POR TESTE - concentre-se em uma única habilidade do SUT por teste
(SUT = System under testing)

↳ ARRANGE, ACT, ASSERT - sempre divida o teste em 3 partes (TRIPLE A)

↳ DIVIDA OS TESTES POR COMPORTAMENTO - não assegure comportamentos distintos em um único teste

↳ RECORRA A GERADORES - use geradores de teste para destacar apenas o que importa no teste (de forma inteligente!)

OBS: Embora os testes não garantam a ausência de defeitos, desempenham um papel essencial para elevar a qualidade do produto //

◣ Desenvolvimento Seguro (OWASP)

↳ Ao abordar a questão da segurança na qualidade de software, são levantadas discussões a respeito de CONFIDENCIALIDADE, INTEGRIDADE, AUTENTICIDADE e NÃO REPÚDIO //

✱ Confidencialidade

↳ garante que todos os dados e serviços são protegidos contra acesso não autorizado

↳ acesso à informação apenas às partes interessadas

↳ atenção às leis nacionais e internacionais (LGPD, HIPAA, GDPR...)

↳ LEI GERAL DE PROTEÇÃO DE DADOS

✱ Integridade

↳ garante que os dados ou serviços não estão sujeitos a manipulação não autorizada //

↳ mecanismos de verificação de integridade (checksums, hashs, assinaturas digitais)

✱ Não Repúdio

↳ garante que o remetente de uma mensagem não pode negar ter enviado e que o destinatário não pode negar ter recebido

↳ criptografia de chave pública e certificados digitais //

✱ Autenticidade

↳ garante que ambas as partes de uma transação são quem elas dizem ser

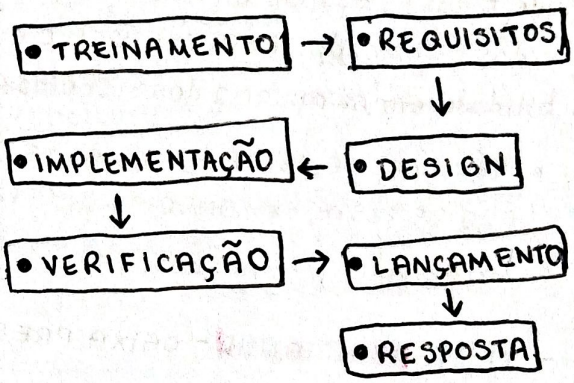
↳ autenticação: 2FA e MFA
↳ autenticação em 2 fatores (senha e token)

✱ Rastreabilidade do Uso

↳ garante o rastreamento de atividades de forma que o sistema possa reconstruí-la

↳ políticas de segurança e conformidade

✱ Ciclo de Vida do Desenvolvimento seguro - SDL



- CICLO DE VIDA DO DESENVOLVIMENTO DE SOFTWARE SEGURO
  ↳ conjunto de princípios de design e boas práticas a serem implementadas no SDLC para detectar, prevenir e corrigir defeitos de segurança no desenvolvimento e aquisição de aplicativos //
  ↳ ciclo de vida de desenvolvimento de software //

SRE (engenharia de confiabilidade) → DESIGN
↓
TESTES ← CÓDIGO
↓
DEPLOY

* DevSecOps
↳ DESENVOLVIMENTO, SEGURANÇA e OPERAÇÕES
↳ extensão da prática de DevOps
↳ cada termo define diferentes funções e responsabilidades das equipes de software durante a ~~aplicação~~ criação de aplicações de software //
abordagem da engenharia de software que aproxima os desenvolvedores de software e os operadores do sistema //
↳ exemplo de implementações:

• GUIDELINE - OWASP

SECRET SCANNING → SCA → SAST
↳ verificação de possíveis vazamentos de credenciais
↳ análise de composição de software
teste de segurança estático

IAC SCANNING ← CONTAINER SCANNING ← DAST
escaneamento de infraestrutura do projeto
escaneamento de container (Ex: docker)
teste dinâmico de segurança

INFRA SCANNING → COMPLIANCE CHECK
verificação de vulnerabilidade de infraestrutura
auditoria de conformidade

OBS: FERRAMENTAS PARA ESSA GUIDELINE:
git secret, git lake, sonarcloud, sonarqube, etc.
↳ integrados por CENTRAL VULNERABILITY MANAGEMENT //

OBS2: O QUE É O OWASP?
↳ o Open Web Application Security Project é uma organização sem fins lucrativos focada em melhorar a segurança de software

* Principais Riscos de Segurança de Aplicações Web - Top 10 da OWASP

• BROKEN ACCESS CONTROL
↳ violação do princípio de privilégios na aplicação
↳ acesso a endpoints de APIs sem controle de acesso
↳ alteração de contas de terceiros por ID exclusivo

• FALHAS DE CRIPTOGRAFIA
↳ algum dado é revelado por texto não criptografado
↳ algoritmo ou protocolo criptográfico antigos
↳ criptografia não aplicada no HTTP //

• INJEÇÃO
↳ os dados fornecidos pelo usuário não são validados, filtrados ou transformados pela aplicação
↳ dados hostis são usados em parâmetros de pesquisa de mapeamento objeto-relacional (ORM) para extrair registros confidenciais adicionais.

• DESIGN NÃO SEGURO
↳ falhas críticas de design e arquitetura em aplicações
↳ falta de controles de segurança integrados à aplicação durante todo o ciclo de desenvolvimento //

• CONFIGURAÇÃO INCORRETA DE SEGURANÇA
↳ recursos desnecessários são ativados ou instalados (por exemplo portas, serviços, páginas, contas ou privilégios desnecessários)
↳ O tratamento de erros revela rastreamentos de pilhas ou outras mensagens de erros excessivamente informativas aos usuários //

• COMPONENTES DATADOS E VULNERÁVEIS
↳ se o software estiver vulnerável, sem suporte ou desatualizado. Incluindo sistema operacional, servidor, SGBD, API, etc.
↳ se não houver a verificação de vulnerabilidades regularmente //

- IDENTIFICAÇÃO E AUTENTICAÇÃO DE FALHAS
  - → permite ataques automatizados, como preenchimento de credenciais em que o invasor possui uma lista de senhas e usuários válidos //
    - → permite ataques de "força bruta"
    - → permite senhas padrão, fracas ou conhecidas

- FALHAS DE SOFTWARE E INTEGRIDADE DE DADOS
  - → aplicação depende de plug-ins, bibliotecas ou módulos de fontes, repositórios e redes de entrega de conteúdo (CDNs) não confiáveis
  - → ADERIR A FUNCIONALIDADE DE ATUALIZAÇÃO AUTONÁTICA! → RISCOS

- FALHAS DE MONITORANENTO E SEGURANÇA DE LOG-IN
  - → eventos auditáveis, como logins, logins com falha e transações de alto valor não são registrados
    - → avisos e erros geram mensagens de log inexistentes, inadequadas ou pouco claras //

- FALHAS DE REQUISIÇÃO DO SERVIDOR
  - → ocorrem sempre que uma aplicação web busca um recurso remoto sem validar o URL fornecido pelO usúrio
  - → permite que um invasor coaja a aplicação a enviar uma solicitação elaborada para um destino inesperado //

* Benefícios do Desenvolvimento Seguro
  - → REDUÇÃO DE VULNERABILIDADES E FALHAS DE SEGURANÇA
    - → AUMENTO DE CONFIABILIDADE DO SOFTWARE
    - → PROTEÇÃO CONTRA ATAQUES
    - → MELHORIA DA IMAGEM E REPUTAÇÃO DA EMPRESA //