Conceitos Básicos de Segurança de Software

Design de segurança, Modelagem de ameaças, Codificação segura, Testes de segurança e Privacidade.

Design de Segurança

O que é

- Trata-se da prática de adotar medidas de segurança desde a fase de concepção do software.
- Ou seja, o software deve possuir mecanismos de proteção quando se olha para o seu funcionamento como um todo, desde a arquitetura até a relação entre suas diferentes funcionalidades.
- Pode-se averiguar a segurança analisando apenas a modelagem do software.

Alguns conceitos fundamentais

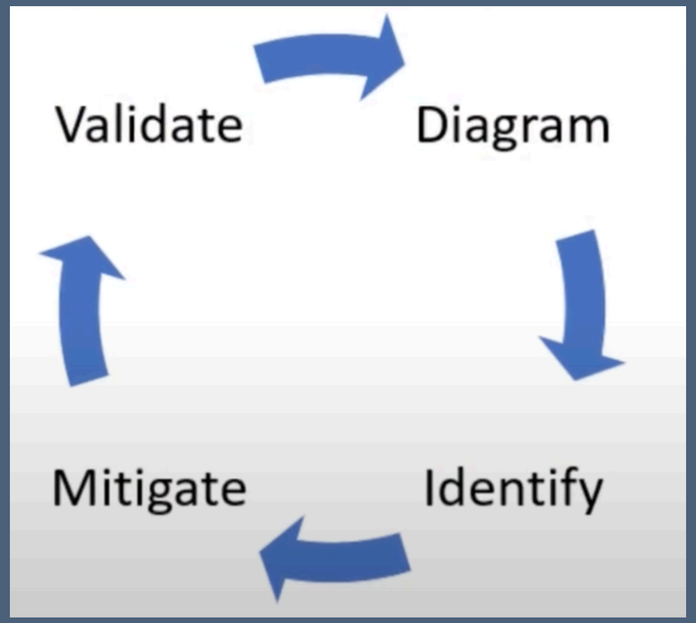
- **Princípio do menor privilégio:** cada componente ou usuário do sistema deve possuir apenas as permissões estritamente necessárias para desempenhar suas funções.
  - Exemplos: usar JWT's com limitações de acordo com o usuário e com tempo de expiração; criar contas no banco de dados para limitar as operações que podem ser feitas nele de acordo com o papel do usuário; limitar as operações que podem ser feitas via API; a UI da aplicação deve ser específica para cada papel de usuário; etc.
- **Defesa em profundidade:** uso de múltiplas camadas de segurança para dificultar ataques.
  - Exemplos: Autenticação (login) multi-fator; Separar autenticação de autorização; Criptografia; etc.
- **Redução da superfície de ataque:** trata-se de reduzir ao máximo o número de pontos de acesso do sistema e a exposição de informações sensíveis.
  - Exemplos: limitar interfaces externas (como API's); proteger painéis de administrador (restringir acesso, limitar papéis e permissões, definir url's não muito óbvias, login com 2FA e captcha); atentar para bibliotecas desatualizadas, antigas e com falhas conhecidas; remover dependências não utilizadas; não armazenar chaves em código; não expor informações no tratamento de erros (exemplo: erro no login não pode informar se determinado e-mail está ou não registrado); etc.

Modelagem de Ameaças

O que é

- Trata-se de uma atividade onde se busca identificar pontos sensíveis do sistema, que poderiam ser usados de forma maliciosa ou que poderiam causar a exposição de dados sensíveis em caso de falhas.

- Alguns exemplos de modelagens de ameaças: STRIDE, PASTA, OCTAVE, etc.



Etapas

- **Identificação de ativos críticos:** definir quais dados armazenados e processados pela aplicação são sensíveis de serem expostos e precisam de proteção. Exemplos: dados pessoais, dados privados, etc.

- **Identificação de ameaças:** pensar nos diferentes ataques que podem comprometer esses dados.

- **Avaliação de riscos:** avaliar a probabilidade e o impacto para cada ameaça identificada, de forma a priorizar os esforços de mitigação.

- **Desenvolvimento de mitigação:** Definir e implementar a proteção para cada ameaça identificada.

O Modelo STRIDE

- Cada letra representa um tipo de ameaça:
  - **S: Spoofing:** fingir ser outra pessoa ou sistema, violando autenticação.

    Como mitigar: Autenticação Multi-Fator, Certificados digitais, etc.
  - **T: Tampering:** modificar e alterar dados sem autorização, violando integridade.

    Como mitigar: Monitoramento constante de arquivos, Sanitização de dados de entrada, etc.
  - **R: Repudiation:** negar a autoria de determinada ação, violando não-repudiação.

    Como mitigar: Implementar logs para ações sensíveis, Implementar assinaturas digitais, etc.
  - **I: Information Disclosure:** expor informações sensíveis, violando confidencialidade.

    Como mitigar: Criptografia, Controle de Acesso, etc.
  - **D: Denial of Service:** sobrecarregar ou interromper serviços, violando disponibilidade.

    Como mitigar: Implementar filtros de tráfego, Implementar balanceamento de cargas, etc.
  - **E: Elevation of Privilege:** obter acesso não autorizado a funções de níveis superiores, violando autorização

    Como mitigar: Implementar o princípio do menor privilégio

Codificação Segura

O que é

- Trata-se de um conjunto de medidas de segurança que podem ser tomadas a nível de código.

Exemplos

- **Validação de entradas:** garantir que as entradas recebidas de fontes externas (principalmente, mas não se limitando a usuários) sejam tratadas e verificadas de forma a evitar ataques como:
  - Injeção SQL: manipulação/alteração de uma consulta SQL a partir de dados de entrada em aplicações. Exemplo de precaução: Sanitizar entradas com aspas, ponto e vírgula e palavras como “admin”.
  - Estouro de buffer: causar o estouro de memória buffer com o objetivo de corromper uma aplicação. Exemplo de precaução: Limitar o tamanho da entrada de acordo com o buffer utilizado.
  - Erros aritméticos inteiros: causar erros em operações de números inteiros, como por exemplo: causar divisão por zero, Overflow (entrada maior do que suportado pelo tipo) ou truncamento (perda de precisão).
  - Cross-site Scripting: Injeção de código malicioso em aplicações web. Exemplo de precaução: Sanitizar entradas com “>”, “<” e “&”.
- **Criptografia:** utilizar criptografia em repouso (dados sensíveis devem ser armazenados em banco de forma criptografada); criptografia em trânsito (usar https e bibliotecas como openSSL); proteger chaves; etc.
- **Gerenciamento seguro de senhas:** criptografar senhas ou utilizar serviços seguros e consolidados de terceiros.
- **Gerenciamento seguro de erros e logs:** remover “prints” de código em produção; tratar bem os erros; não fornecer informações sensíveis ao se comunicar com usuários; etc.

Testes de Segurança

O que é

- São um conjunto de atividades que buscam verificar e validar a segurança de um software.
- Diferentemente dos testes funcionais, não buscam verificar “apenas” se um sistema está funcionando de acordo com suas especificações. Buscam, além disso, verificar se o sistema produzido está fazendo isso de forma segura e correta.
- Através da avaliação de riscos, busca-se verificar os riscos que permeiam determinado sistema de acordo com suas especificação e construção.
- Tipos de teste:
  - Análise estática de código
  - Análise dinâmica de código
  - Testes de penetração
  - Testes de injeção

Privacidade

O que é

- Trata-se de um direito humano de ter suas informações pessoais e privadas protegidas e não violadas.
- Todo sistema deve garantir esse direito, sob pena de punição legal ou perda de reputação.
- Tipos de dados importantes de privacidade:
  - Informações pessoais: nome completo, CPF ou outro documento de identificação, data de nascimento, endereço, telefone, profissão, escolaridade, relações de parentesco, etc.
  - Informações privadas: condições médicas, registros de comunicações, sigilo bancário e financeiro, etc.

Práticas Recomendadas de Design de Privacidade

- “Privacy by Design”: a proteção da privacidade não é um requisito não funcional e adicional. É um requisito FUNCIONAL e PRIMÁRIO.

    1. Proatividade, não reatividade: antecipar e prevenir problemas de privacidade antes que aconteçam, em vez de apenas consertar erros que possam vir a ocorrer.

    2. Privacidade como padrão: o usuário não deve solicitar, ativar ou configurar a proteção de sua privacidade. Este aspecto deve ser padrão e regra para todos os usuários.

    3. Minimização: deve-se solicitar e salvar o mínimo de dados possíveis dos usuários.

    4. Proteção: os dados que devem ser armazenados e processados devem ser protegidos com o máximo de empenho.

    5. Transparência: deve-se deixar claro para o usuários quais dados são armazenados e com qual propósito.