**CHECKLIST DE PULL REQUEST**

**1. O Portão: Entradas e Saídas**

Toda entrada externa é um vetor de ataque em potencial. Toda saída é uma oportunidade de vazamento.

**[ ] 1.1 - Validação de Entrada no Backend**

* + **Check**: Toda nova entrada (APIs, formulários, parâmetros de URL) é rigorosamente validada no backend?
  + **Risco**: Injeção de código (SQLi, XSS, Command Injection).
  + **Prática Segura**: Utilize listas de permissão (allow-lists) para tipos de dados, formatos e faixas de valores. Nunca confie apenas no frontend.
  + **Referência**: Cap. 10.2 (A03: Injection), Cap. 12.3 (C3: Validate All Input).

**[ ] 1.2 - Codificação de Saída Contextual**

* + **Check**: Os dados são codificados corretamente para o contexto (HTML, atributos, JavaScript, CSS)?
  + **Risco**: Cross-Site Scripting (XSS).
  + **Prática Segura**: Use auto-escaping do framework e, quando necessário, bibliotecas de codificação específicas.
  + **Referência**: Cap. 10.2 (A03: Injection — XSS).

**2. As Chaves do Castelo: Autenticação e Autorização**

Uma falha aqui pode entregar o controle total do sistema a um adversário.

**[ ] 2.1 - Autorização Explícita e Centralizada**

* + **Check**: A lógica de autorização está no backend e é aplicada de forma consistente?
  + **Risco**: Broken Access Control.
  + **Prática Segura**: Centralize a lógica em middlewares ou decoradores reutilizáveis.
  + **Referência**: Cap. 10.2 (A01), Cap. 11 (API01, API05).

**[ ] 2.2 - Verificação de Nível de Objeto (Propriedade)**

* + **Check**: Ao acessar um recurso específico, verifica-se se o usuário autenticado é o dono ou tem permissão?
  + **Risco**: BOLA / IDOR.
  + **Prática Segura**: Inclua validação explícita (ex.: AND dono\_id = ?).
  + **Referência**: Cap. 10.2 (A01), Cap. 11 (API01).

**3. O Cofre: Gestão de Segredos e Dados Sensíveis**

Segredos no código são uma bomba-relógio.

**[ ] 3.1 - Ausência de Segredos Codificados (Hardcoded)**

* + **Check**: O PR está livre de senhas, tokens, chaves de API?
  + **Risco**: Comprometimento de sistemas e vazamento de dados.
  + **Prática Segura**: Utilize gerenciadores de segredos e injete-os em tempo de execução.
  + **Referência**: Cap. 10.2 (A05), Cap. 17 (Seção 17.4.7).

**[ ] 3.2 - Logging Seguro**

* + **Check**: Os logs estão livres de dados sensíveis (senhas, PII, tokens)?
  + **Risco**: Vazamento de informações críticas e violações de privacidade.
  + **Prática Segura**: Filtre e mascare dados sensíveis nos logs.
  + **Referência**: Cap. 10.2 (A09).

**4. A Fundação: Cadeia de Suprimentos**

Não se constrói um castelo sobre areia movediça.

**[ ] 4.1 - Veto de Novas Dependências**

* + **Check**: Novas bibliotecas ou pacotes são confiáveis, mantidos e livres de CVEs críticos?
  + **Risco**: Vulnerabilidades e ataques à cadeia de suprimentos.
  + **Prática Segura**: Integre análise SCA na pipeline. Verifique também escopo e permissões que novas dependências exigem (ex.: OAuth scopes, permissões de APIs).
  + **Referência**: Cap. 10.2 (A06), Cap. 14.