**UNIVERSIDADE TUIUTI DO PARANÁ**

**JULIANA BASTOS RITTEL**

**ANDERSON ALVES DOS SANTOS**

# PROGRAMAÇÃO MOBILE

**CURITIBA**

**2025**

**JULIANA BASTOS RITTEL**

**NICKSON JEAN FERREIRA WALACHY**

# PROGRAMAÇÃO MOBILE

Relatório para a disciplina de Programação Mobile, da Universidade Tuiuti do Paraná, com o objetivo de ampliar conhecimentos sobre programação mobile. Como requisito avaliativo do 2° do bimestre.

Professor: Chaua Coluene Queirolo Barbosa da Silva.

**CURITIBA**

**2025**

## RESUMO

Trata-se de um trabalho que investiga os principais desafios de segurança no desenvolvimento de aplicativos móveis, com foco nas vulnerabilidades mais comuns encontradas em plataformas Android e iOS. O objetivo é analisar como práticas inadequadas, permissões mal gerenciadas e falhas na proteção de dados podem comprometer a privacidade e a integridade das informações dos usuários. A pesquisa explora soluções técnicas e preventivas, como uso de criptografia, autenticação segura, armazenamento protegido e controle de permissões, além de destacar ferramentas de análise e monitoramento. Também são apresentados casos reais de falhas em apps populares, evidenciando a importância da segurança no ciclo de vida do software. Por fim, é proposto um checklist prático para desenvolvedores, visando garantir a proteção dos dados desde a concepção até a publicação do aplicativo.



Palavras-chave: Segurança mobile; Desenvolvimento seguro; Criptografia; Permissões de apps; Privacidade de dados.

**SUMÁRIO**

# 1 INTRODUÇÃO

A presente pesquisa visa explorar os desafios e práticas relacionadas à segurança no desenvolvimento de aplicativos móveis, com ênfase nas plataformas Android e iOS. Este trabalho delimita seu foco na análise das principais vulnerabilidades que afetam apps móveis e nas estratégias técnicas e preventivas adotadas para mitigar esses riscos. Os objetivos centrais deste estudo são investigar as metodologias e boas práticas utilizadas para garantir a proteção dos dados dos usuários, além de compreender o impacto da gestão adequada de permissões, autenticação e armazenamento seguro na segurança geral dos aplicativos.

A motivação para esta pesquisa surge da crescente utilização de dispositivos móveis para atividades sensíveis, como transações financeiras e armazenamento de informações pessoais, que exigem elevados níveis de segurança. A adoção de soluções robustas para proteção de dados, bem como a conscientização sobre os riscos inerentes ao desenvolvimento mobile, são elementos-chave para evitar vulnerabilidades que possam comprometer a privacidade e a integridade dos usuários. Além disso, a relevância de estudar casos reais de falhas de segurança e o uso de ferramentas de análise torna-se fundamental para a construção de aplicativos confiáveis e seguros.

Os tópicos principais que serão abordados no desenvolvimento deste trabalho incluem: as principais vulnerabilidades em apps Android e iOS, boas práticas de segurança como criptografia e autenticação, o gerenciamento de permissões e privacidade do usuário, casos reais de falhas em apps populares, ferramentas e técnicas de análise de segurança, além de um checklist prático para desenvolvedores visando garantir a segurança no ciclo de vida do aplicativo.

## 2 VULNERABILIDADE EM APPS ANDROID E IOS

Antes de aplicar boas práticas de segurança, é fundamental conhecer as vulnerabilidades mais comuns que afetam aplicativos móveis. Tanto no Android quanto no iOS, diversas falhas podem comprometer a integridade dos dados e a confiança dos usuários.

A seguir, são apresentadas as principais ameaças enfrentadas durante o desenvolvimento mobile, que servem de base para a definição de estratégias eficazes de mitigação.

1. Misconfiguração de segurança
    a. Permissões excessivas, debug ativado em produção, credenciais hardcoded, atividades/content providers expostos.
2. Armazenamento inseguro de dados
    a. Dados sensíveis em SharedPreferences, plists, cache não criptografado.
3. Comunicação vulnerável (MitM)
    a. Ausência de HTTPS/TLS, sem certificate pinning.
4. Código pouco protegido
    a. Sem obfuscação, sem checagem de integridade, vulnerável a hooks/emuladores.
5. Autenticação fraca
    a. Senhas inseguros, sem autenticação multifator.
6. Dependências e bibliotecas desatualizadas
    a. Introduzem vulnerabilidades conhecidas.

## 3 BOAS PRÁTICAS DE SEGURANÇA

Após identificar os riscos mais frequentes, o próximo passo é aplicar boas práticas que reforcem a segurança do aplicativo desde as primeiras linhas de código. A adoção de medidas preventivas — como criptografia, autenticação robusta e gestão adequada de permissões — é essencial para proteger os dados do usuário e reduzir vulnerabilidades. Esta seção apresenta técnicas e recomendações práticas que devem ser incorporadas ao ciclo de vida do desenvolvimento mobile.

1. Criptografia
    a. Utilize algoritmos robustos como AES 256 para criptografar dados armazenados e em trânsito.
    b. Aplique TLS 1.2 ou superior em todas as conexões e implemente certificate pinning.
    c. No Android, use o Keystore com suporte a hardware (StrongBox); no iOS, utilize o Keychain e, se possível, o Secure Enclave.
    d. Bancos de dados locais devem ser criptografados com soluções como SQLCipher.
2. Autenticação
    a. Implemente autenticação multifator (MFA) e biometria sempre que possível.
    b. Use algoritmos seguros para armazenamento de senhas (como bcrypt ou Argon2).
    c. Gerencie sessões com tokens seguros, expiração automática e revogação de sessão.
3. Armazenamento seguro
    a. Evite armazenar dados sensíveis em arquivos de configuração, cache ou preferências padrão.
    b. Utilize APIs de armazenamento seguro como Android Keystore ou iOS Keychain.
    c. Limpe dados sensíveis do dispositivo ao realizar logout ou desinstalar o aplicativo.
4. Permissões e privacidade

    a. Solicite permissões apenas no momento necessário e com explicações claras para o usuário.
    b. Aplique o princípio do menor privilégio, evitando acesso desnecessário a sensores ou dados.
    c. Utilize mecanismos seguros de compartilhamento de dados entre apps, como content URIs com permissões controladas.
5. Código e integridade
    a. Obfusque o código-fonte para dificultar engenharia reversa.
    b. Implemente verificações de integridade para detectar modificações não autorizadas.
    c. Detecte e bloqueie execução em ambientes comprometidos, como dispositivos com root ou jailbreak.
6. Dependências e atualizações
    a. Mantenha todas as bibliotecas e SDKs atualizados.
    b. Verifique regularmente a presença de vulnerabilidades conhecidas nas dependências.
    c. Planeje ciclos de atualização contínuos e mecanismos de atualização forçada em caso de falhas críticas.
7. Monitoramento e resposta
    a. Utilize ferramentas de monitoramento e logs seguros para detectar comportamentos suspeitos.
    b. Estabeleça um plano de resposta a incidentes e canal para recebimento de relatos de falhas de segurança.
    c. Integre práticas de segurança ao pipeline de desenvolvimento, com validações automatizadas.

## 4 PERMISSÕES E PRIVACIDADE DO USUÁRIO

A privacidade do usuário é um dos pilares da segurança em aplicativos móveis. Muitos riscos de segurança surgem não por falhas técnicas diretas, mas pelo uso inadequado de permissões ou pela coleta excessiva de dados pessoais. Aplicativos que pedem acesso irrestrito à câmera, microfone, localização ou contatos sem justificativa clara comprometem a confiança e expõem o usuário a riscos desnecessários. Por isso, a gestão criteriosa de permissões e o respeito às diretrizes de privacidade são fundamentais para qualquer app que preze pela segurança e transparência.

Boas práticas em permissões e privacidade:

- Solicite permissões apenas quando necessário: evite pedir acesso a funcionalidades que o aplicativo não utiliza diretamente. A solicitação deve ocorrer no contexto da ação, e não ao abrir o app pela primeira vez.
- Justifique o uso de cada permissão: forneça ao usuário uma explicação clara sobre por que aquela permissão está sendo pedida e como ela será usada.
- Implemente permissões em tempo de execução: no Android, utilize a abordagem de runtime permissions, garantindo que o usuário possa aceitar ou negar cada acesso de forma consciente.
- Ofereça alternativas: sempre que possível, permita que o usuário utilize funcionalidades do app mesmo sem conceder todas as permissões — com funcionalidades limitadas, se necessário.
- Gerencie permissões dinamicamente: respeite mudanças feitas pelo usuário nas configurações do sistema e adapte o funcionamento do app de acordo com as permissões concedidas ou revogadas.
- Use APIs de acesso seguro a dados sensíveis: como FileProvider no Android para compartilhamento controlado de arquivos, evitando a exposição de diretórios internos.
- Evite coleta excessiva de dados: limite a coleta de informações pessoais ao estritamente necessário para a operação do app. Desative qualquer rastreamento que não tenha finalidade clara para o usuário.

- Ofereça controles de privacidade: permita que o usuário veja, gerencie e exclua seus dados sempre que desejar, em conformidade com legislações como LGPD e GDPR.

## 5 CASOS REAIS DE FALHAS EM APPS POPULARES

Estudar incidentes de segurança que afetaram aplicativos reais é uma das formas mais eficazes de compreender as consequências práticas das vulnerabilidades discutidas. Esses casos demonstram como erros aparentemente simples — como má configuração de permissões ou falhas de criptografia — podem levar a vazamentos de dados massivos, violações de privacidade e danos à reputação das empresas. A seguir, são apresentados exemplos marcantes que ilustram os riscos de negligenciar a segurança no desenvolvimento mobile.

Exemplos de falhas reais em aplicativos

Aplicativos de relacionamento e estilo de vida:

- Aplicativos de nicho, como plataformas de namoro e fetiches, já foram responsáveis por vazamentos de milhões de imagens e dados pessoais devido a bancos de dados e buckets de armazenamento mal configurados, acessíveis publicamente sem autenticação.

Apps com geolocalização ativa:

- Aplicativos como Strava, que mapeiam rotas de corrida, acidentalmente revelaram a localização de bases militares e rotinas de soldados, devido à exposição de dados agregados de forma pública sem filtros adequados de segurança.

Aplicativos bancários falsos em lojas oficiais:

- Diversos apps foram identificados como versões fraudulentas de bancos reais, simulando interfaces legítimas para capturar credenciais dos usuários. Muitos chegaram a ser publicados na Play Store antes de serem removidos.

Coleta oculta de dados de localização:

- Aplicativos populares, como jogos e apps de produtividade, foram flagrados vendendo dados de localização para empresas de publicidade e análise, sem o devido consentimento dos usuários ou de forma disfarçada nos termos de uso.

Vulnerabilidades em apps de comunicação:

- Explorações do tipo "zero-click", como a que afetou o WhatsApp, permitiram a instalação de spyware apenas com o recebimento de uma chamada, sem qualquer ação do usuário. Esse tipo de falha destaca a importância de validações profundas em canais de entrada.

Exposição de APIs internas:

- Algumas empresas deixaram endpoints internos sem autenticação ou com tokens de acesso públicos embutidos no código do aplicativo, permitindo que terceiros manipulassem ou extraíssem dados do sistema backend.

## 6 FERRAMENTAS E TÉCNICAS DE ANÁLISE DE SEGURANÇA MOBILE

Para garantir que um aplicativo móvel esteja verdadeiramente seguro, não basta aplicar boas práticas durante o desenvolvimento — é essencial validá-las com ferramentas e técnicas específicas. Testes estáticos, dinâmicos e de integridade permitem identificar vulnerabilidades antes que cheguem ao usuário final. Além disso, soluções de monitoramento ajudam a detectar comportamentos anômalos em tempo real. A seguir, estão listadas as principais abordagens e ferramentas utilizadas na análise de segurança mobile, cobrindo todas as fases do ciclo de vida de um app.

Técnicas e ferramentas recomendadas:

Análise estática (SAST): avalia o código-fonte ou os binários sem executar o aplicativo, identificando problemas de lógica, más práticas e possíveis vazamentos de dados.

Ferramentas:

- SonarQube
- Checkmarx
- Semgrep
- MobSF (Mobile Security Framework)

Análise dinâmica (DAST): executa o app em tempo real e observa seu comportamento frente a ataques, manipulação de inputs e interceptação de comunicação.

Ferramentas:

- Burp Suite
- OWASP ZAP
- Frida (instrumentação de apps)
- Appium + scripts de automação com ataques simulados

Testes de penetração: simula ataques reais ao aplicativo e à infraestrutura associada, com foco em descobrir falhas que não seriam detectadas apenas com ferramentas automatizadas.

Realizados manualmente por especialistas ou com apoio de frameworks como:

- Metasploit
- Needle (iOS)
- Drozer (Android)

Verificação de integridade: garante que o app não foi modificado ou executado em ambientes comprometidos, protegendo contra engenharia reversa e ataques por dispositivos com root/jailbreak.

Técnicas:

- Checagem de hash de assinatura
- Detecção de emuladores
- Validação via APIs nativas (ex: Android SafetyNet, Play Integrity, iOS App Attest)

Monitoramento e resposta: ferramentas de observabilidade são fundamentais para rastrear erros, tentativas de violação e falhas em produção.

Soluções recomendadas:

- Sentry, Firebase Crashlytics (para exceções e falhas)
- Datadog, New Relic (para performance e comportamento)
- WAF e SIEM em backend (para detecção e resposta a ameaças)

Integração no ciclo de desenvolvimento (DevSecOps)

- Integração de scanners no pipeline CI/CD
- Gatilhos automáticos de análise a cada build ou pull request
- Automação de correções com base em alertas de segurança

## 7 CHECKLIST DE SEGURANÇA PARA DESENVOLVEDORES

Mesmo com conhecimento técnico e boas intenções, falhas de segurança podem surgir se não houver um processo sistemático de verificação. Um checklist claro e objetivo ajuda desenvolvedores a não esquecerem etapas cruciais de segurança antes de publicar ou atualizar um aplicativo. Esta seção apresenta um guia prático que pode ser usado como referência em revisões de código, testes finais e auditorias internas, garantindo que o app atenda aos principais requisitos de proteção de dados, privacidade e integridade.

Permissões:

- Solicitar apenas as permissões estritamente necessárias ao funcionamento do app.
- Realizar a solicitação de permissões em tempo de execução, e não na instalação.
- Justificar claramente ao usuário o motivo de cada permissão solicitada.
- Permitir que o app funcione, com limitações, mesmo se o usuário negar certas permissões.

Criptografia:

- Utilizar TLS 1.2 ou superior para toda comunicação com servidores.
- Implementar certificate pinning para proteger contra ataques Man-in-the-Middle.
- Criptografar dados locais com algoritmos fortes, como AES-256.
- Armazenar chaves e tokens sensíveis usando o Android Keystore ou iOS Keychain.

Autenticação:

- Exigir senhas fortes e únicas por usuário.
- Suportar autenticação multifator (MFA) ou biometria.
- Gerenciar sessões com tempo de expiração, logout automático e revogação de tokens.

Armazenamento:

- Não armazenar dados sensíveis em cache, arquivos temporários ou preferências padrão.
- Limpar todos os dados sensíveis durante logout ou desinstalação.
- Garantir o uso de armazenamento seguro e criptografado para qualquer informação crítica.

Código e integridade:

- Obfuscar o código-fonte antes de gerar o build de produção.
- Implementar validação de integridade para detectar modificações no app.
- Detectar e impedir execução em dispositivos com root, jailbreak ou emuladores.

Dependências:

- Utilizar apenas bibliotecas confiáveis, mantidas e com histórico de segurança.
- Manter todas as dependências atualizadas com frequência.
- Verificar e mitigar vulnerabilidades conhecidas em bibliotecas de terceiros.

Monitoramento:

- Implementar ferramentas de monitoramento de falhas e comportamento do app.
- Analisar logs e eventos para detectar comportamentos suspeitos.
- Evitar o registro de informações sensíveis nos logs.

Testes e auditoria:

- Realizar testes estáticos de segurança no código-fonte (SAST).
- Executar testes dinâmicos e simulações de ataque (DAST, pentest).
- Integrar verificações de segurança no pipeline de CI/CD.

Privacidade e conformidade legal:

- Exibir uma política de privacidade clara e acessível ao usuário.
- Permitir que o usuário acesse, edite ou exclua seus próprios dados.
- Estar em conformidade com leis de privacidade como a LGPD e GDPR.

Distribuição segura:

- Assinar o aplicativo com uma chave segura e protegida.

- Distribuir apenas por meios oficiais como Google Play ou App Store.
- Garantir que builds de produção estejam livres de configurações de teste ou debug.

## 8 CONCLUSÃO

Este trabalho teve como objetivo explorar os principais desafios e práticas relacionadas à segurança no desenvolvimento de aplicativos móveis, com foco nas plataformas Android e iOS. Discutiu-se como vulnerabilidades comuns, má gestão de permissões e práticas inseguras podem comprometer a privacidade e a integridade dos dados dos usuários. Foram analisadas boas práticas como criptografia, autenticação segura e armazenamento protegido, além de casos reais que ilustram as consequências da negligência na segurança.

A pesquisa também destacou a importância do uso de ferramentas de análise e testes contínuos para identificar e mitigar riscos ao longo do ciclo de vida do aplicativo. A principal dificuldade foi a constante atualização das técnicas e ameaças, exigindo uma postura proativa e um compromisso contínuo dos desenvolvedores com a segurança.

Conclui-se que a proteção efetiva dos dados em aplicativos móveis depende de uma abordagem integrada que combine boas práticas técnicas, conscientização sobre permissões e privacidade, além de monitoramento constante. Assim, é possível garantir a confiança dos usuários e a integridade das plataformas digitais frente ao cenário atual de ameaças.

## REFERÊNCIAS

OWASP Mobile Security Testing Guide. Disponível em: <https://owasp.org/www-project-mobile-security-testing-guide/>. Acesso em: 01 jul. 2025.

OWASP. OWASP Mobile Top 10. Disponível em: <https://owasp.org/www-project-mobile-top-10/>. Acesso em: 01 jul. 2025.

Security. Disponível em: <https://developer.apple.com/security/>. Acesso em: 01 jul. 2025.