UNIVERSIDADE TUIUTI DO PARANÁ

ALEXANDRE JUNIOR GARCIA DOS SANTOS
ÍCARO DE ASSIS HOSTERT
LEONARDO DE PAULA TEIXEIRA
VITOR ROBERTO BATISTA SCHIRMER

Segurança em Aplicativos Mobile: Riscos, Vulnerabilidades e Boas Práticas

CURITIBA

2025

ALEXANDRE JUNIOR GARCIA DOS SANTOS
ÍCARO DE ASSIS HOSTERT
LEONARDO DE PAULA TEIXEIRA
VITOR ROBERTO BATISTA SCHIRMER

Segurança em Aplicativos Mobile: Riscos, Vulnerabilidades e Boas Práticas

Relatório apresentado ao Curso de Análise e Desenvolvimento de Sistemas da Universidade Tuiuti do Paraná como requisito avaliativo do 5º bimestre da disciplina de Desenvolvimento de sistemas.

Professores: Chaua coluene queirolo barbosa da silva

CURITIBA

2025

RESUMO

Este estudo aborda a criticidade da segurança no desenvolvimento de aplicativos móveis para as plataformas Android e iOS, diante da crescente utilização destes para o manuseio de dados sensíveis. O objetivo central é analisar os principais riscos e vulnerabilidades de segurança, apresentando um conjunto de boas práticas e soluções técnicas para sua mitigação. A pesquisa explora as dez principais vulnerabilidades catalogadas pelo OWASP Mobile Top 10, detalha práticas essenciais de segurança como criptografia de dados em trânsito e em repouso, autenticação forte e armazenamento seguro, além de discutir a importância da gestão de permissões para a privacidade do usuário. O trabalho também analisa casos reais de falhas de segurança, apresenta ferramentas de análise estática e dinâmica (SAST/DAST) e compila um checklist prático para desenvolvedores. Conclui-se que a segurança mobile é um processo contínuo que deve ser integrado desde a concepção do software (*Security by Design*), sendo um pilar fundamental para a confiança do usuário e o sucesso do produto.

Palavras-chave: Desenvolvimento Seguro; OWASP; Vulnerabilidades de Aplicativos; Privacidade de Dados.

Introdução

Com o crescimento exponencial do uso de aplicativos móveis para atividades sensíveis como transações financeiras, troca de mensagens e armazenamento de dados pessoais, a segurança tornou-se um fator crítico no desenvolvimento mobile. A conveniência oferecida por estes aplicativos acarreta uma superfície de ataque igualmente vasta. Vulnerabilidades comuns, má configuração de permissões e práticas inseguras de programação podem comprometer seriamente a privacidade, a integridade e a disponibilidade dos dados dos usuários, resultando em perdas financeiras, roubo de identidade e quebra de confiança.

Este estudo propõe uma investigação aprofundada sobre os principais riscos de segurança no ecossistema de desenvolvimento mobile, com foco nas plataformas Android e iOS. O objetivo central é analisar os desafios técnicos e apresentar um conjunto de soluções preventivas e boas práticas que devem ser integradas ao ciclo de vida de desenvolvimento de software. A pesquisa abrange desde as vulnerabilidades mais recorrentes, catalogadas por entidades como a OWASP (Open Web Application Security Project), até a análise de ferramentas, casos de falhas notórios e a compilação de um checklist prático para desenvolvedores, visando fortalecer a segurança dos aplicativos e proteger seus usuários finais.

Desenvolvimento

Principais Vulnerabilidades em Apps Android e iOS A OWASP mantém um projeto focado nos dez principais riscos de segurança em aplicações móveis, o *OWASP Mobile Top 10*. Este ranking é a principal referência do setor. As vulnerabilidades mais críticas incluem:

1. **Uso Inadequado de Credenciais:** Armazenamento de senhas, tokens de sessão ou chaves de API de forma insegura no dispositivo (ex: em SharedPreferences ou UserDefaults sem criptografia), permitindo que um invasor com acesso ao sistema de arquivos as extraia.
2. **Qualidade Insuficiente do Código:** Erros de lógica de programação, manipulação inadequada de memória ou a falta de validação de entradas que podem levar a comportamentos inesperados e exploráveis.
3. **Comunicação Insegura:** Transmissão de dados sensíveis (login, dados pessoais, informações financeiras) através de canais não criptografados (HTTP em vez de HTTPS). Isso expõe os dados a ataques de interceptação (*Man-in-the-Middle*).

4. **Autenticação Insegura:** Falhas no processo de verificação de identidade do usuário. Exemplos incluem permitir senhas fracas, não proteger contra ataques de força bruta ou gerenciar sessões de forma inadequada (tokens que nunca expiram).
5. **Criptografia Insuficiente:** Utilizar algoritmos de criptografia fracos, obsoletos ou implementá-los de forma incorreta, o que torna os dados supostamente protegidos fáceis de serem decifrados.
6. **Autorização Insegura:** Permitir que um usuário autenticado acesse recursos ou execute ações para as quais não deveria ter permissão. Exemplo: um usuário comum acessar funcionalidades de administrador.
7. **Proteção Insuficiente do Lado do Cliente:** Falta de ofuscação do código, ausência de detecção de *root* (Android) ou *jailbreak* (iOS), e falta de proteção contra a depuração em tempo de execução, facilitando a engenharia reversa do aplicativo.
8. **Manipulação de Código:** Capacidade de um invasor modificar o código do aplicativo para introduzir funcionalidades maliciosas, como roubo de dados, e distribuí-lo em lojas não oficiais.
9. **Engenharia Reversa:** Facilidade com que um invasor pode analisar o binário do aplicativo para entender sua lógica de funcionamento, encontrar segredos (chaves, senhas) e identificar vulnerabilidades.
10. **Funcionalidade Extrânea:** Manter código de depuração, logs excessivos ou endpoints de teste ocultos na versão de produção, que podem ser descobertos e explorados por invasores.

Boas Práticas de Segurança

Criptografia de Dados

A criptografia é fundamental para proteger a confidencialidade dos dados. Ela se divide em duas categorias principais:

- **Dados em Trânsito (Data in Transit):** Toda a comunicação entre o aplicativo e seus servidores deve ocorrer exclusivamente sobre o protocolo **TLS (Transport Layer Security)**, idealmente na sua versão mais recente (1.2 ou superior). Isso é implementado utilizando o protocolo HTTPS. Práticas como *SSL Pinning* podem ser usadas para evitar ataques de interceptação, garantindo que o app se comunique apenas com o servidor legítimo.
- **Dados em Repouso (Data at Rest):** Dados sensíveis armazenados no dispositivo (bancos de dados, arquivos, preferências) devem ser criptografados. O padrão recomendado é o **AES-256**. As chaves de criptografia, por sua vez, devem ser gerenciadas de forma segura, utilizando os mecanismos nativos de cada plataforma, como o **Keychain** no iOS e o **Keystore** no Android.

Autenticação e Gerenciamento de Sessão

- **Autenticação Forte:** Nunca armazene senhas em texto claro. Utilize algoritmos de *hashing* robustos e com *salt* (como Argon2, scrypt ou bcrypt) no lado do servidor.
- **Autenticação Multifator (MFA):** Ofereça e incentive o uso de um segundo fator de autenticação (ex: código via SMS, app autenticador como Google Authenticator, ou notificações push).

- **Autenticação Biométrica:** Utiliza as APIs nativas (**Face ID/Touch ID** no iOS e **BiometricPrompt** no Android) para permitir que o usuário acesse o app ou confirme ações críticas de forma segura e conveniente.
- **Gerenciamento de Sessão:** Use tokens de sessão (como JWTs - JSON Web Tokens) que sejam de curta duração e invalide-os no servidor após o logout. Implemente mecanismos de *refresh token* para renovar a sessão sem exigir que o usuário faça login novamente a todo momento.

Armazenamento Seguro

A escolha do local de armazenamento depende da sensibilidade do dado:

- **Dados Altamente Sensíveis (Tokens, Chaves de API, Senhas):** Utilizar **Keychain Services** (iOS) e **Android Keystore System**. Esses são contêineres de hardware seguros projetados para armazenar pequenas quantidades de dados criptográficos.
- **Dados de Usuário (Arquivos, Banco de Dados):** Armazenar em diretórios privados do aplicativo e aplicar criptografia de arquivo ou de banco de dados (ex: SQLCipher).
- **Dados Não Sensíveis (Configurações, Cache):** Podem ser armazenados em UserDefaults (iOS) ou SharedPreferences (Android), mas nunca para guardar informações sigilosas.

Permissões de Apps e a Privacidade do Usuário

O **Princípio do Menor Privilégio** é a regra de ouro: um aplicativo deve solicitar apenas as permissões estritamente necessárias para seu funcionamento.

- **Justificativa Clara:** Ao solicitar uma permissão (câmera, localização, contatos), explique ao usuário de forma clara e concisa por que o acesso é necessário. O iOS exige que essa justificativa seja incluída no arquivo Info.plist, e o Android recomenda exibi-la antes da solicitação.
- **Solicitação em Contexto:** Peça a permissão apenas no momento em que o recurso for ser utilizado pela primeira vez, e não todas de uma vez na inicialização do app. Isso aumenta a confiança do usuário.
- **Auditoria Regular:** Revise periodicamente as permissões solicitadas e remova aquelas que não são mais necessárias em novas versões do aplicativo.

Casos Reais de Falhas de Segurança

1. **Zoom (2020):** A popularidade do aplicativo durante a pandemia expôs diversas falhas. Uma delas foi a alegação de usar criptografia de ponta a ponta, quando na verdade a comunicação era apenas criptografada via TLS padrão, permitindo que a própria Zoom tivesse acesso às chaves das reuniões. Outra falha no cliente macOS permitia que um invasor obtivesse acesso de *root* ao sistema.
2. **British Airways (2018):** O aplicativo móvel e o site da companhia aérea foram comprometidos por um ataque que injetou um script malicioso (Magecart) nas páginas de pagamento. Este script capturou dados de mais de 380.000 transações com cartão de crédito, incluindo nomes, e-mails e os números completos dos cartões, com data de validade e código CVV. A falha residia na segurança inadequada de suas aplicações web, que eram consumidas pelo app.

3. **Grindr (2018):** Foi revelado que o aplicativo de relacionamento estava transmitindo dados de status de HIV dos usuários, juntamente com sua localização e e-mail, para empresas terceiras de otimização de software. Os dados eram enviados em pacotes que poderiam, em alguns casos, serem interceptados, representando uma grave violação de privacidade e segurança da comunicação.

Ferramentas e Técnicas de Análise de Segurança Mobile

A análise de segurança pode ser automatizada (SAST/DAST) ou manual (pentest).

- **Análise Estática de Segurança de Aplicações (SAST):** Ferramentas que analisam o código-fonte ou o binário do aplicativo sem executá-lo, em busca de padrões de vulnerabilidades conhecidas.
  - **Exemplos:** MobSF (Mobile Security Framework), SonarQube, Checkmarx.
- **Análise Dinâmica de Segurança de Aplicações (DAST):** Ferramentas que analisam o aplicativo em tempo de execução, monitorando seu comportamento, tráfego de rede, chamadas de sistema e interações com o sistema de arquivos.
  - **Exemplos:** Frida (framework de instrumentação dinâmica), Burp Suite e OWASP ZAP (proxies de interceptação de tráfego), Drozer.
- **Teste de Invasão (Penetration Testing - Pentest):** Um processo manual e aprofundado onde um especialista em segurança (hacker ético) tenta ativamente explorar as vulnerabilidades de um aplicativo, simulando um ataque real para avaliar a postura de segurança de forma completa.

Checklist de Segurança para Desenvolvedores

Este checklist resume as principais ações que um desenvolvedor deve seguir:

- **Arquitetura e Design**: Adotar o Princípio do Menor Privilégio para permissões. Planejar o armazenamento seguro de cada tipo de dado (Keychain/Keystore para segredos). Modelar ameaças para identificar potenciais vetores de ataque no início do projeto.
- **Comunicação de Rede**:Utilizar HTTPS (TLS 1.2+) para TODA a comunicação de rede. Implementar SSL Pinning para aplicações de alto risco (financeiras, saúde). Não transmitir dados sensíveis nos parâmetros da URL.
- **Autenticação e Sessão** Implementar autenticação multifator (MFA).Utilizar as APIs biométricas nativas de forma segura.Garantir que os tokens de sessão tenham tempo de expiração curto. Implementar um mecanismo de logout que invalide o token no servidor.
- **Armazenamento Local**:NÃO armazenar senhas, tokens ou chaves de API em texto claro ou em locais inseguros .Criptografar bancos de dados e arquivos sensíveis armazenados no dispositivo. Limpar o cache e dados sensíveis da memória quando o app vai para o background.
- **Código e Build**:Validar e higienizar todas as entradas do usuário e de fontes externas (APIs).Utilizar ferramentas SAST e DAST durante o desenvolvimento e a integração contínua. Ofuscar o código (ProGuard/R8 no Android, ferramentas de terceiros no iOS). Remover todo o código de depuração, logs excessivos e endpoints de teste das versões de produção. Implementar detecção de root/jailbreak para apps de alto risco.

Conclusão

A segurança em aplicativos móveis é um processo contínuo e multifacetado, não um objetivo final. Como demonstrado neste estudo, as ameaças são variadas, indo desde falhas técnicas na implementação de criptografia e comunicação até decisões de design que violem a privacidade do usuário. A referência do OWASP Mobile Top 10 serve como um guia essencial para que as equipes de desenvolvimento priorizem seus esforços na mitigação dos riscos mais críticos.

A adoção de uma mentalidade de *Security by Design* (Segurança desde a Concepção) é a abordagem mais eficaz. Isso significa integrar as boas práticas de segurança, como o uso rigoroso de criptografia, autenticação forte, gerenciamento seguro de armazenamento e a aplicação do princípio do menor privilégio, em todas as fases do ciclo de vida do desenvolvimento. O uso de ferramentas de análise estática e dinâmica, aliado a testes de invasão periódicos, complementa o processo, garantindo a identificação e correção de vulnerabilidades antes que possam ser exploradas.

Em última análise, a segurança de um aplicativo não é apenas uma responsabilidade técnica, mas também um pilar da confiança do usuário. Em um mercado competitivo, a capacidade de proteger os dados e a privacidade dos usuários é um diferencial crucial que impacta diretamente a reputação e o sucesso do produto.

Referências Bibliográficas

ANDROID DEVELOPER DOCUMENTATION. Security. Disponível em: https://developer.android.com/topic/security. Acesso em: 05 jun. 2025.

APPLE DEVELOPER DOCUMENTATION. Security. Disponível em: https://developer.apple.com/documentation/security. Acesso em: 05 jun. 2025.

GOOGLE PLAY ACADEMY. App permissions best practices. Disponível em: https://play.google.com/academy/courses/app-permissions-best-practices/. Acesso em: 05 jun. 2025.

OWASP MOBILE SECURITY PROJECT. OWASP Mobile Security Project. Disponível em: https://owasp.org/www-project-mobile-security/. Acesso em: 05 jun. 2025.

OWASP MOBILE APPLICATION SECURITY VERIFICATION STANDARD (MASVS). OWASP Mobile Application Security Verification Standard (MASVS). Disponível em: https://github.com/OWASP/owasp-masvs. Acesso em: 05 jun. 2025.