Guia de Requisitos de Segurança e Privacidade para Aplicativos Móveis

LEI GERAL DE PROTEÇÃO DE DADOS PESSOAIS (LGPD)

Versão 1.0
Brasília, novembro de 2021

GUIA DE REQUISITOS DE SEGURANÇA E PRIVACIDADE PARA APLICATIVOS MÓVEIS

MINISTÉRIO DA ECONOMIA

Paulo Roberto Nunes Guedes

Ministro

SECRETARIA ESPECIAL DE DESBUROCRATIZAÇÃO, GESTÃO E GOVERNO DIGITAL

Caio Mario Paes de Andrade

Secretário Especial de Desburocratização, Gestão e Governo Digital

SECRETARIA DE GOVERNO DIGITAL

Fernando Andre Coelho Mitkiewicz

Secretário de Governo Digital

DEPARTAMENTO DE GOVERNANÇA DE DADOS E INFORMAÇÕES

Leonardo Rodrigo Ferreira

Diretor do Departamento de Governança de Dados e Informações

COORDENAÇÃO-GERAL DE SEGURANÇA DA INFORMAÇÃO

Loriza Andrade Vaz de Melo

Coordenadora-Geral de Segurança da Informação

Equipe Técnica de Elaboração

Álvaro Sergio de Souza Junior

Amaury C. da Silveira Junior

Bruno Pierre Rodrigues de Sousa

Eder Ferreira de Andrade

Francisco Magno Felix Nobre

Heráclito Ricardo Ferreira Gomes

Ivaldo Jeferson de Santana Castro

Raphael César Estevão

Yuri Arcanjo de Carvalho

Equipe Revisora

Marcelo de Lima

Marcus Paulo Barbosa Vasconcelos

Histórico de Versões

Data	Versão	Descrição	Autor
24/11/2021	1.0	Primeira versão do Guia de Requisitos Mínimos de Segurança da Informação e Privacidade para Aplicativos Móveis.	Equipe Técnica de Elaboração

SUMÁRIO

INTRODUÇÃO 7

DEFINIÇÕES GERAIS 5

1 Privacidade e proteção de dados em aplicativos móveis 9

1.1 Privacidade e proteção de dados em aplicativos móveis 9

1.2 Estratégias de design de privacidade 11

2 Identificação e proteção de dados sensíveis 17

2.1 Proteção de dados sensíveis em dispositivos móveis 17

2.2 Armazenamento e processamento de dados 18

2.3 Exclusão de dados 20

2.4 Evitando exposição de dados 20

2.5 Proteção de dados sensíveis em trânsito 23

2.6 Incrementando segurança no tráfego de dados 233

2.7 Utilização de autenticação multifator 24

3 Implementação de autenticação de usuário, autorização e gerenciamento de sessão segura 25

3.1 Implementando controles de autenticação e autorização 25

4 Gerenciamento de fatores de autenticação e autorização com segurança no dispositivo 28

4.1 Utilização de Tokens 28

4.2 Armazenamento de senhas 29

5 Proteção de back-end, do servidor de plataforma e APIs 30

5.1 Implementando segurança em APIs ..... **Erro! Indicador não definido.**

6 Integração segura de dados com código de terceiros ..... **Erro! Indicador não definido.**1

6.1 Utilizando código de terceiros 32

7 Consentimento e proteção de privacidade 32

7.1 Coletando dados pessoais 33

7.2 Obtendo consentimento 34

7.3 Auditoria na coleta de dados do usuário 37

8 Proteção para recursos de pagamento 37

8.1 Registro dos recursos de pagamentos 38

8.2 Precauções com recursos de pagamento 38
8.3 Controle de acesso 39
9 Distribuição segura de software 39
9.1 Publicação do aplicativo 39
10 Interpretação do tempo de execução de código 41
10.1 Orientações para segurança de interpretadores de códigos 41
11 Verificação da integridade do dispositivo e do aplicativo 41
11.1 Integridade do dispositivo 42
11.2 Desativando os recursos do desenvolvedor 42
12 Proteção do aplicativo contra injeções do lado do usuário 43
12.1 Reduzindo o risco de ataques por injeção de código 43
13 Garantindo o uso correto de sensores biométricos e hardware seguro 45
13.1 Utilizando recursos biométricos com segurança 45
REFERÊNCIAS BIBLIOGRÁFICAS 48

INTRODUÇÃO

Baseado no artigo 46 da LGPD, o Guia de Segurança e Privacidade para Aplicativos Móveis tem propósito de fomentar orientações básicas aos desenvolvedores de aplicativos móveis da Administração Pública Federal sobre o desenvolvimento seguro das aplicações móveis.

Diretrizes relativas ao desenvolvimento de códigos seguros (ex.: Ciclo de Vida de Desenvolvimento de Software) e para proteção de servidores (ex.: Defesa em Profundidade) ainda são válidas, e devem ser empregadas quando pertinentes. Neste guia, serão identificadas propriedades e funções específicas dos aplicativos móveis para ajudar a leitora ou o leitor a construir aplicativos móveis mais seguros e confiáveis.

Os requisitos presentes neste guia não possuem caráter obrigatório, tampouco constituem itens exaustivos. Por este motivo, ficará a cargo da equipe de desenvolvimento identificar os requisitos do aplicativo a ser desenvolvido.

Destaca-se que este guia se destina à aplicativos móveis que são desenvolvidos para smartphones e tablets. Os exemplos dados ao longo do Guia são para plataformas IOS e Android. Dispositivos vestíveis, Internet das Coisas (IoT) e outros sistemas operacionais não fazem parte do escopo desse guia.

Ressalta-se que o presente documento constitui um complemento aos demais guias operacionais já elaborados e publicados pela Secretaria de Governo Digital da Secretaria Especial de Desburocratização, Gestão e Governo Digital do Ministério da Economia. Tais guias estão disponíveis na página oficial da SGD na Internet.

DEFINIÇÕES GERAIS

Para auxílio na leitura do guia, serão adotadas as seguintes definições no que se refere a incidentes ocorridos nos órgãos da administração pública federal.

Air-gap:

Medida de segurança de rede empregada em um ou mais computadores que garante o isolamento físico de redes não seguras, como a Internet pública ou uma rede local não segura.

Android:

Sistema operacional móvel baseado em uma versão modificada do kernel Linux e outro software de código aberto, projetado principalmente para dispositivos móveis com tela de toque, como smartphones e tablets.

API:

Acrônimo de Interface de Programação de Aplicações (Application Programming Interface).

App store:

Loja de aplicativos. Em português, é um portal online por meio do qual programas de software são disponibilizados para os usuários, seja de forma paga ou gratuita.

Autenticação:

Processo que busca verificar a identidade digital de uma entidade de um sistema de informação quando tal entidade requisita acesso. O processo é realizado por meio de regras preestabelecidas, geralmente pela comparação das credenciais apresentadas pela entidade com outras já pré-definidas no sistema de informação, reconhecendo como verdadeiras ou legítimas as partes envolvidas em um processo.

Autenticação de multifatores (MFA):

Utilização de dois ou mais fatores de autenticação para concessão de acesso a um sistema. Os fatores de autenticação se dividem em: algo que o usuário conhece (senhas, frases de segurança, PIN, dentre outros); algo que o usuário possui (certificado digital, tokens, códigos enviados por SMS, dentre outros); algo que o usuário é (aferível por meios biométricos, tais como digitais, padrões de retina, reconhecimento facial, dentre outros); e onde o usuário está (quando o acesso só pode ser feito em uma máquina específica, cujo acesso é restrito).

Autorização:

Processo que ocorre após a autenticação e tem a função de diferenciar os privilégios atribuídos ao usuário que foi autenticado. Os atributos de autorização normalmente são definidos em grupos mantidos em uma base de dados centralizada, sendo que cada usuário herda as características do grupo a que ele pertence. Portanto, autorização é o direito ou permissão de acesso a um recurso de um sistema de informação.

Back-end:

Parte de um site ou programa de software que os usuários não veem (camada de acesso a dados).

Criptografia:

Arte de proteção da informação através de sua transformação em um texto cifrado (criptografado), com o uso de uma chave de cifragem e de procedimentos computacionais previamente estabelecidos, a fim de que somente o(s) possuidor(es) da chave de decifragem possa(m) reverter o texto criptografado de volta ao original (texto pleno). A chave de decifragem pode ser igual (criptografia simétrica) ou diferente (criptografia assimétrica) da chave de cifragem.

Criptografia assimétrica:

É qualquer sistema criptográfico que usa pares de chaves: chaves públicas, que podem ser amplamente disseminadas, e chaves privadas que são conhecidas apenas pelo proprietário. Isto realiza duas funções: autenticação, em que a chave pública verifica que um portador da chave privada parelhada enviou a mensagem, e encriptação, em que apenas o portador da chave privada parelhada pode decriptar a mensagem encriptada com a chave pública.

Cookies:

Cookies são pequenos arquivos que são gravados no computador quando acessa sites na Internet e que são reenviados a estes mesmos sites quando novamente visitados. São usados para manter informações sobre o usuário, como carrinho de compras, lista de produtos e preferências de navegação.

Dado pessoal:

Informação relacionada a uma pessoa natural identificada ou identificável.

Dado Pessoal Sensível:

Dado pessoal sobre origem racial ou étnica, convicção religiosa, opinião política, filiação a

sindicato ou a organização de caráter religioso, filosófico ou político, dado referente à saúde ou à vida sexual, dado genético ou biométrico, quando vinculado a uma pessoa natural.

Deny-list:

É um mecanismo de controle de acesso básico que nega todos os elementos nela contidos, exceto aqueles não mencionados.

DoS:

Tipo de ataque cibernético no qual o ofensor busca tornar uma máquina ou recurso de rede indisponível para seus usuários, interrompendo temporariamente ou indefinidamente os serviços do host alvo.

Front-end:

Parte de um site ou programa de software que se refere à interface de usuário (camada de apresentação).

iOS:

Sistema operacional móvel para dispositivos fabricados pela Apple.

LGPD:

Lei nº 13.709, de 14 de agosto de 2018 – Lei Geral de Proteção de Dados Pessoais (LGPD), cujo objetivo é proteger os direitos fundamentais de privacidade e de liberdade de cada indivíduo

Mod:

Abreviação de Modification (ou Modificação, em português). É usado para dar nomes a softwares modificados com o intuito de fazer melhorias, desbloqueio de funcionalidades ou criação de um novo software baseado no código fonte do original.

Privacy by default:

Princípio segundo o qual as configurações de privacidade mais restritivas devem ser aplicadas por padrão em um produto ou serviço que foi lançado ao público.

Privacy by design:

Princípio segundo o qual a privacidade deve ser incorporada a um sistema, produto ou serviço durante todo o seu ciclo de vida.

Opt-in:

Processo pelo qual o usuário autoriza uma determinada ação por parte de uma organização, geralmente o tratamento de dados e o seu compartilhamento com organizações parceiras, ou o recebimento de mensagens enviadas por organizações.

Opt-out:

Processo pelo qual o usuário desautoriza uma empresa a continuar com uma determinada ação previamente permitida.

Allow-list:

É um mecanismo de controle de acesso básico que permite passar todos os elementos nela contidos, exceto aqueles não mencionados.

QoS:

Quality of Service ou Qualidade de Serviço, em português, é um conjunto de tecnologias que funciona em uma rede para assegurar sua capacidade de executar tráfego prioritário e aplicativos, de forma confiável e com capacidade de rede limitada.

Security by default:

Metodologia em que as configurações de segurança mais restritivas possíveis devem ser aplicadas por padrão a um produto ou serviço que foi lançado ao público.

Security by design:

Abordagem segundo a qual a segurança dos dados é incorporada a um sistema, produto ou serviço durante todo o seu ciclo de vida.

Walled garden control:

Técnica de segurança de rede que visa impedir o acesso de pessoas com ou sem determinados MODs.

Webview:

Recurso que pode exibir o conteúdo interativo da web e carregar strings HTML.

1 Privacidade e proteção de dados em aplicativos móveis

Privacidade é um tópico de pesquisa ativa e contínua. Considerando as práticas e complexidades atuais de desenvolvimento de aplicativos, as estratégias de design de privacidade combinadas com os objetivos de proteção de dados pessoais podem tornar esse processo mais compreensível.

A figura abaixo ilustra as estratégias de design de privacidade propostas neste guia.

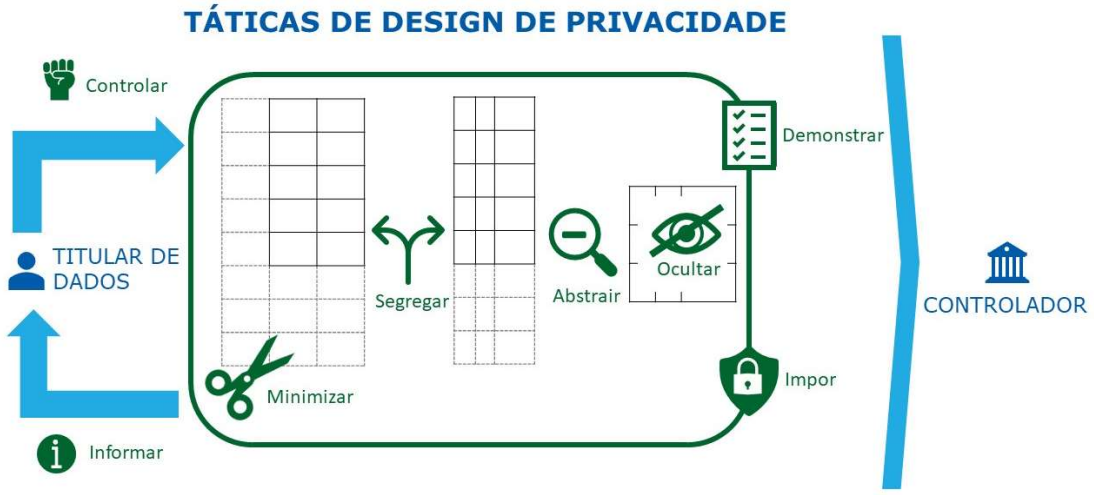


Figura 1 - Estratégias de design de privacidade

Os objetivos de proteção de dados e as táticas de design de privacidade apontadas a seguir apresentam requisitos e opções em uma linguagem de fácil compreensão. As estratégias apresentadas abaixo expõem soluções para a melhoria do cenário nacional vigente, no qual as garantias de privacidade e proteção de dados pessoais raramente são incorporadas.

1.1 Metodologia privacy by design em aplicativos móveis

O objetivo dessa metodologia é fornecer uma visão geral das melhores práticas referentes a padrões de design de privacidade. A partir dos exemplos abaixo, ficará demonstrado que mesmo decisões arquitetônicas aparentemente pequenas podem ter uma grande influência na privacidade e na proteção de dados.

Exemplo disso é o fato de que o aplicativo deve ser capaz de identificar possíveis abusos de acesso: ao invés de coletar identificadores do dispositivo (IDs de hardware), que podem expor desnecessariamente o usuário, pode coletar informações de data, hora e localização. De maneira ilustrativa, essas informações possivelmente seriam suficientes para que se conseguisse verificar

um mesmo login sendo utilizado no Brasil e, em pouco tempo, ser acessado na China, configurando então um possível cenário de violação de acesso. Tal alteração cumpriria o resultado de segurança que se quer atingir, sem comprometer a privacidade do indivíduo.

Outra decisão a ser tomada é sobre onde os dados pessoais serão armazenados. Por exemplo, se o local de hospedagem está em um país em que os dados poderiam ser acessados sem justificativa por um governo estrangeiro, tal acesso poderia colocar em risco a privacidade dos indivíduos. Há de se considerar também como tais dados podem ser protegidos contra hackers, ou no caso de perda dos dispositivos móveis.

Os desenvolvedores de aplicativos móveis devem avaliar os aspectos de segurança e privacidade ao utilizar componentes de terceiros, pois seu comportamento pode representar riscos de privacidade e segurança para os usuários, por exemplo, coletando dados do usuário por conta própria, sem base legal e transparência para os usuários. A arquitetura deve facilitar a troca de componentes de terceiros se descobrir que estes não se comportam como prometido e afrontem os termos da Lei Geral de Proteção de Dados Pessoais - LGPD.

A decisão sobre qual funcionalidade está conectada (por exemplo, comunicação sempre criptografada) e o que pode ser configurável pelo usuário influencia o grau de privacidade e proteção de dados. Para todas as configurações, o desenvolvedor deve decidir se há uma pré-configuração, ou se é solicitado ao usuário no momento da instalação. A Lei Geral de Proteção de Dados Pessoais – LGPD exige que qualquer configuração siga o princípio de "proteção de dados" (Artigo 6º), de modo que a quantidade de dados pessoais coletados, a extensão do seu processamento, o período de seu armazenamento e sua acessibilidade são limitados ao necessário para cada propósito específico. No caso específico, se as funcionalidades de rastreamento ou personalização não forem necessárias para o propósito, a configuração padrão deve garantir que os respectivos dados não sejam processados, a menos que o usuário assuma os riscos e altere ativamente a configuração.

Os riscos de privacidade e proteção de dados dos aplicativos móveis decorrem principalmente de duas dimensões:

a) sua natureza, como software executado em dispositivos móveis privados (dispositivos portáteis), e;

b) as particularidades do ambiente de desenvolvimento e distribuição móvel como tal.

A seguir, compartilham-se algumas estratégias de design vinculado aos objetivos de privacidade e proteção de dados.

1.2 Estratégias de design de privacidade

Uma **estratégia de design de privacidade**[1] envolve a arquitetura de um sistema de informação que compreenda essa temática desde sua concepção, objetivando atingir um nível adequado de proteção à privacidade.

A descrição das estratégias de design de privacidade frequentemente envolve o tratamento de dados pessoais. Os desenvolvedores devem estar cientes de que o conceito legal de tratamento é amplo: deve incluir coleta, produção, recepção, classificação, utilização, acesso, reprodução, transmissão, distribuição, processamento, arquivamento, armazenamento, eliminação, avaliação ou controle da informação, modificação, comunicação, transferência, difusão ou extração de dados pessoais, no que couber.

As **oito estratégias de design de privacidade**[2] e suas respectivas táticas são descritas a seguir com exemplos de como podem ser aplicadas no desenvolvimento de um aplicativo móvel.

Estratégia 1 – Minimizar: Limitar ao máximo o processamento de dados pessoais.

Táticas associadas	
Supressão	Abster-se de processar os dados pessoais de um titular de dados, parcial ou totalmente, da mesma forma que incluir na deny-list.
Seleção	Decidir sobre quais dados pessoais serão utilizados, selecionando apenas aquilo que for estritamente necessário à sua finalidade.
Elmina	Eliminar logicamente (deletar, apagar) os dados pessoais não mais necessários e determinar previamente o tempo de utilização para que o dado seja excluído.
Descarte	Destruir, inclusive dos backups, os dados pessoais não mais necessários de maneira segura (irreversível), de modo a impossibilitar a recuperação.

[1] M. Colesky, J.-H. Hoepman e C. Hillen, "A Critical Analysis of Privacy Design Strategies," in International Workshop on Privacy Engineering - IWPE'16, San Jose, CA, EUA, 2016.

[2] J. H. Hoepman., "Privacy Design Strategies," in IFIP TC11 29th Int. Conf. sobre Segurança da Informação (IFIP SEC 2014), 2014.

Exemplo da estratégia Minimizar:

Os aplicativos devem limitar seu acesso aos sensores (localização, movimento, câmera, microfone) e dados armazenados localmente (fotos, contatos) ao mínimo e apenas quando claramente relevantes para o funcionamento adequado do aplicativo. Um aplicativo que peça desnecessariamente a permissão para acessar todos esses recursos é o exemplo perfeito do que não ser feito.

No entanto, um aplicativo de clima que solicite a localização do usuário para fornecer informações sobre o clima local pode ser aceitável. No entanto, observe que, para que um aplicativo de tempo forneça uma previsão meteorológica, ele não precisa saber a localização exata do usuário. Em vez disso, basta uma indicação aproximada (por exemplo, o nome da cidade ou a indicação da área em vários quilômetros quadrados).

Estratégia 2 - Segregar: Evitar a correlação de dados pessoais, separando o processamento lógico e físico.

Táticas associadas	
Distribuição	Particionar dados pessoais para que seja necessário mais acesso para processá-los.
Isolamento	Processar partes de dados pessoais de forma independente, sem acesso ou correlação com partes relacionadas.

Exemplo da estratégia Segregar:

A ampliação dos recursos de rede (e a permissão de uso de dados) tornam os aplicativos ponto a ponto, em que os usuários compartilham ou se comunicam diretamente sem a ajuda de um servidor central.

Os dispositivos móveis são poderosos em termos de processamento, largura de banda e armazenamento e, portanto, podem realizar muitas tarefas localmente que eram impensáveis há vários anos. Por exemplo, o reconhecimento de imagens dentro de fotos pode ser feito no smartphone, de forma que o upload de fotos para um servidor central não seja mais necessário.

Estratégia 3 - Abstrair: Limitar ao máximo possível o detalhamento (granularidade e especificidade) dos dados pessoais que estão sendo processados.

Táticas associadas	
Resumo	Extrair semelhanças em dados pessoais, encontrando e processando correlações ao invés dos próprios dados.
Agrupamento	Induzir menos detalhes de dados pessoais antes do processamento, alocando-os em categorias comuns.
Perturbação	Adicionar ruído ou aproximar o valor real de um item de dados.

Exemplo da estratégia Abstrair:

Os serviços baseados em localização geralmente precisam apenas de uma indicação aproximada da localização do usuário atual para oferecer uma visão geral dos serviços próximos a esse local. Portanto, ao invés de usar as coordenadas GPS precisas oferecidas pelo smartphone, um aplicativo baseado em localização poderia considerar um local mais abrangente antes de consultar os serviços associados de um servidor central. Os dispositivos móveis podem disponibilizar vários níveis de granularidade de localização, como chamadas de API do sistema operacional, e até mesmo permitir que os usuários concedam ou neguem acesso a dados de localização mais precisos por aplicativo.

Uma abordagem mais genérica para autenticação amigável de privacidade é quando as pessoas têm acesso aos dados com base em atributos mais gerais (por exemplo, se o usuário tem uma assinatura), ao invés de usar a identidade da pessoa para tomar essa decisão de acesso. O chamado suporte de Credenciais Baseadas em Atributos (ou ABCs, do inglês attribute-based credentials) oferece suporte a isso de uma maneira desvinculada e com privacidade amigável – o que faz com que os ABCs também se enquadrem na estratégia de ocultar, descrita abaixo.

Estratégia 4 - Ocultar: Proteger dados pessoais ou torná-los desvinculáveis ou inobserváveis. Evitar que dados pessoais se tornem públicos. Evitar a exposição de dados pessoais restringindo o acesso ou ocultando sua própria existência.

Táticas associadas	
Restrição	Prevenir o acesso não autorizado aos dados pessoais.
Embaralhamento	Processar dados pessoais aleatoriamente dentro de um grupo grande o suficiente para reduzir a correlação.
Criptografia	Criptografar dados (em trânsito ou em repouso).
Ofuscação	Tornar os dados pessoais incompreensíveis para impedir a capacidade de decifrá-los.
Dissociação	Remover a correlação entre diferentes partes de dados pessoais.

Exemplo da estratégia Ocultar:

> No mínimo, os aplicativos devem criptografar todas as suas comunicações e usar a fixação de certificado (ou chaves pré-instaladas) para evitar ataques *man-in-the-middle*, quando adversários podem comprometer a infraestrutura do certificado TLS. Aplicativos mais avançados tentarão ocultar os metadados usando técnicas de embaralhamento ou implantando uma rede de roteamento em camadas (onion routing, por exemplo, Tor).

Estratégia 5 - Informar: Fornecer aos titulares dos dados informações adequadas sobre quais dados pessoais são processados, como são processados e para qual finalidade.

Táticas associadas	
Entrega	Disponibilizar amplos recursos sobre o processamento de dados pessoais, incluindo políticas, processos e riscos potenciais.
Notificação	Alertar os titulares dos dados sobre qualquer nova informação sobre o processamento de seus dados pessoais em tempo hábil.
Explicação	Detalhar as informações sobre o processamento de dados pessoais de forma concisa e compreensível.

Exemplos da estratégia Informar:

Exemplo 1: Um método possível para comunicar intuitivamente a maneira como um aplicativo lida com dados pessoais, especialmente em virtude da tela pequena de um smartphone, é usar ícones de privacidade. Porém, existe abordagem padronizada atualmente.

Exemplo 2: O acesso a sensores ou dados armazenados localmente pode ser sinalizado para o usuário de maneiras menos intrusivas do que um termo ou um diálogo. Por exemplo, é possível usar ícones especiais em uma barra de status que acendem quando certos tipos de dados confidenciais são acessados e que, quando clicados, fornecem mais informações sobre o acesso específico. Eles dão ao usuário a opção de alterar as configurações para resolver quaisquer questões associadas ao acesso. Como exemplo, pode-se considerar o uso de uma pequena seta na barra de status, na parte superior da tela dos dispositivos iOS, para sinalizar o uso (recente) de serviços de localização.

Estratégia 6 - Controlar: Fornecer aos titulares de dados mecanismos para controlar o processamento dos seus dados pessoais.

Táticas associadas	
Consentimento	Processar apenas os dados pessoais para os quais houve consentimento explícito, dado livremente e recebido.
Escolha granular	Permitir ao usuário selecionar ou excluir, de qualquer processamento, os dados pessoais, parcial ou totalmente.
Atualização	Fornecer aos titulares de dados os meios para manter seus dados pessoais precisos e atualizados.
Retirada	Respeitar o direito do titular dos dados à remoção completa de quaisquer dados pessoais em tempo hábil.

Exemplo da estratégia Controlar:

Ao solicitar permissões para acessar sensores (localização, movimento, câmera, microfone) e dados armazenados localmente (fotos, contatos), os aplicativos móveis ainda devem funcionar (talvez oferecendo funcionalidade limitada) quando esse acesso não é fornecido.

Estratégia 7 - Impor: Comprometer-se com a privacidade no processamento de dados pessoais e se fazer cumpri-la.

Táticas associadas	
Reconhecimento	Reconhecer o valor da privacidade e decidir sobre políticas que a habilitem e processos que respeitem os dados pessoais.
Fundamento	Considerar a privacidade ao projetar ou modificar recursos e atualizar políticas e processos para proteger melhor os dados pessoais.
Dever	Garantir que as políticas sejam respeitadas, tratando os dados pessoais como um ativo, e a privacidade como um objetivo a ser incentivado como um recurso crítico.

Exemplo da estratégia Impor:

Prever o cumprimento à política de privacidade e proteção de dados pessoais do órgão/entidade, nos termos de referência na contratação de um determinado serviço, sob pena de sanções.

É especialmente importante que o desenvolvedor siga a política de privacidade ou de proteção de dados pessoais do órgão/entidade no processo de criação e manutenção de aplicativos.

Estratégia 8 - Demonstrar: Fornecer evidências de que você processa dados pessoais seguindo as melhores práticas de privacidade.

Táticas associadas	
Registro	Rastrear todo o processamento de dados, sem revelar dados pessoais, protegendo e revendo a informação recolhida para quaisquer riscos.
Auditoria	Examinar todas as atividades do dia a dia em busca de quaisquer riscos aos dados pessoais e responder seriamente a quaisquer discrepâncias.
Relatório	Analisar as informações coletadas em testes, auditorias e registros periodicamente para revisar as melhorias na proteção de dados pessoais.

Exemplo da estratégia Demonstrar:

Além das estratégias citadas acima, em que o registro pode ser feito centralmente e no smartphone (e, consequentemente, a auditoria pode ocorrer de ambas as maneiras, talvez por uma ferramenta desenvolvida e fornecida independentemente), a estratégia de "Demonstrar" também demanda que o desenvolvedor do aplicativo selecione cuidadosamente as bibliotecas fornecidas por terceiros, incluídas no aplicativo, para implementar certas funcionalidades. Em particular, deve haver verificação e documentação, de maneira que se possa analisar se a biblioteca não viola a política de privacidade.

Também realizar uma avaliação de impacto da proteção de dados adequada e documentar seu resultado é um fator-chave para essa estratégia.

2 Identificação e proteção de dados sensíveis

Devido à sua natureza portátil, os dispositivos móveis têm maior risco de perda ou roubo. Os desenvolvedores de aplicativos móveis precisam levar isso em consideração e adicionar recursos no seu modelo de segurança, especificamente ao proteger informações sensíveis, tais como dados pessoais do usuário ou dados críticos ao negócio.

Nota: Os dados sensíveis citados neste capítulo não se limitam ao conceito especificado no art. 5º, inciso II da Lei Geral de Proteção de Dados Pessoais (LGPD) de dados pessoais sensíveis. São, portanto, mais abrangentes.

2.1 Proteção de dados sensíveis em dispositivos móveis

Os dispositivos móveis possuem cada vez mais poder de armazenamento e, considerando sua simplicidade e utilidade, é prático que guardem dados sensíveis. Serão apresentadas a seguir algumas recomendações retiradas das melhores práticas conhecidas.

A fase inicial do tratamento dos dados sensíveis é composta por classificar, processar e validar os dados a serem tratados. Isso compreende:

Classificar, processar e validar:

De antemão, identificar quais dados são necessários, sua sensibilidade, e se é apropriado coletar, armazenar e usar cada tipo de dado.

Classificar e aplicar os controles de armazenamento de dados de acordo com a

sensibilidade – por exemplo, senhas, dados pessoais, localização, registros de erros etc.

- **Processar**, armazenar e usar dados de acordo com sua classificação.
- **Validar** a segurança das chamadas de API aplicadas a dados sensíveis.
- Considerar a segurança de todo o ciclo de vida dos dados ao escrever o código do aplicativo: coleta durante a transmissão, armazenamento temporário, armazenamento em cache, backup, exclusão etc.

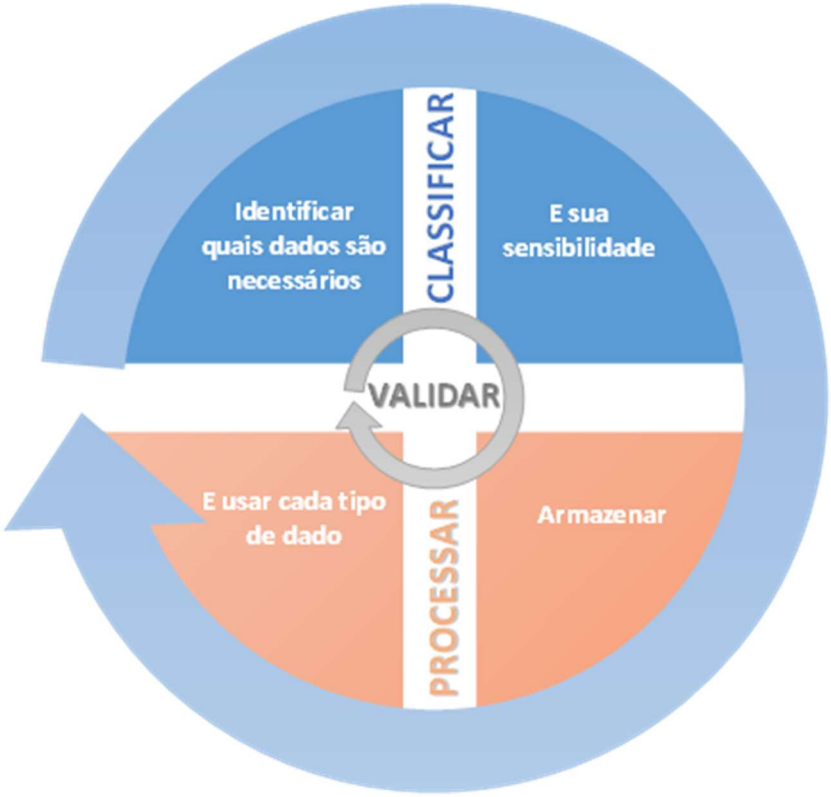


Figura 2 - Recomendações para proteção de dados sensíveis

2.2 Armazenamento e processamento de dados

- Não armazenar dados sensíveis e chaves privadas, a menos que sejam criptografados e, se possível, armazenados de maneira anonimizada em uma área à prova de violação, com suporte pelo fabricante da plataforma.
- Verificar se a criptografia de armazenamento de nível do Sistema Operacional está ativada e o dispositivo está protegido por um PIN ou senha.

- O armazenamento e processamento de dados sensíveis no servidor ou no dispositivo do cliente (usuário) deve ser avaliado caso a caso (segurança relativa do cliente vs. segurança do servidor). Informações mais específicas podem ser encontradas na **OWASP Cloud Top 10**[3] ou na **ENISA Cloud Risk Assessmet**[4] para suporte a decisões.

- Se não for possível evitar o armazenamento de dados sensíveis no dispositivo do usuário, use uma API de criptografia de arquivo fornecida pelo Sistema Operacional ou outra fonte confiável. Algumas plataformas (por exemplo, iOS e Android) fornecem APIs de criptografia de arquivos que usam uma chave secreta protegida pelo código de desbloqueio do dispositivo e exclusão na limpeza remota. Se isso estiver disponível, ele deve ser usado como forma de aumentar a segurança da criptografia sem criar carga extra do lado do usuário final. Além disso, torna os dados armazenados mais seguros em caso de perda ou roubo. No entanto, deve-se ter em mente que, mesmo quando protegido pela chave de desbloqueio do dispositivo, se os dados forem armazenados, sua proteção depende da segurança do código de desbloqueio na hipótese de que a exclusão remota da chave não seja possível por qualquer motivo.

- Não utilizar bibliotecas ou métodos criptográficos da linguagem de programação que estejam obsoletos.

- Não armazenar dados de histórico de localização ou informações críticas à segurança e a privacidade do usuário no dispositivo para além do período exigido pelo aplicativo. Suponha que o armazenamento compartilhado não seja confiável: em tal hipótese, as informações podem facilmente vazar de maneiras inesperadas através de qualquer armazenamento compartilhado. Portanto, recomenda-se:

    - Tratar caches e armazenamento temporário como um possível canal de vazamento, quando compartilhado com outros aplicativos.

    - Tratar o armazenamento compartilhado (e.g. catálogo de endereços, galeria de mídia e arquivos de áudio) como outro possível canal de vazamento. Por exemplo, o armazenamento de imagens com metadados de localização na galeria de mídia permite que as informações sejam compartilhadas de maneiras indesejadas.

[3] OWASP Cloud Top 10 https://www.owasp.org/index.php/Category:OWASP_Cloud_%E2%80%90_10_Project

[4] Computação em nuvem: benefícios, riscos e recomendações para segurança da informação 2009.

2.3 Exclusão de dados

- Para dados pessoais sensíveis, a exclusão deve ser programada obedecendo a sua política de privacidade, com um período máximo de retenção – para evitar, por exemplo, que os dados permaneçam em caches indefinidamente.

  Nota: A legislação acerca do período de retenção deverá ser observada.

- Garantir que, durante a remoção do aplicativo (operação de desinstalação), quaisquer dados sensíveis do usuário e as credenciais específicas do aplicativo correspondentes sejam excluídos do ambiente de execução (data center, servidor etc.), do dispositivo e de qualquer outro meio de armazenamento.

- Os aplicativos em dispositivos gerenciados devem utilizar o **remote wipe (eliminação remota) e Kill Switch APIs**[5] para possibilitar a remoção de informações sensíveis do dispositivo em caso de roubo ou perda.

- Os desenvolvedores de aplicativos podem incorporar um "data kill switch" específico do aplicativo em seus produtos, para permitir a exclusão por app dos dados sensíveis de seu aplicativo. Contudo, uma autenticação forte – como a autenticação multifator, por exemplo – é imprescindível para proteger o uso indevido de tal recurso.

- Excluir os caches do aplicativo no encerramento do aplicativo.

- Os arquivos de banco de dados que contêm dados sensíveis – por exemplo, caches iOS WebView – devem ser removidos manualmente. A exclusão de registros usando a API do banco de dados não levará necessariamente à remoção completa dos dados da estrutura do banco de dados.

2.4 Evitando a exposição de dados

- Aplicar o fundamento do **respeito à privacidade**[6]: Coletar e divulgar apenas os dados necessários para a finalidade do aplicativo (dados identificados na fase de projeto).
- Usar identificadores não persistentes que não são compartilhados com outros aplicativos

[5] "Kill-switch" é o termo usado por um Sistema Operacional com o propósito de remover aplicativos e/ou dados remotamente.
[6] Art. 2º, Inciso I, da Lei nº 13.709, Lei Geral de Proteção de Dados Pessoais http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm. Acesso em 04 mai. 2021.

sempre que possível (por exemplo, não use os identificadores de hardware exclusivos do dispositivo, como IMEI ou UDID como um identificador).

- Não transferir dados protegidos por permissão para outros aplicativos. Isso ocorre quando permissões específicas são necessárias para acessar os dados e, no entanto, um aplicativo que recebeu essas permissões disponibiliza os dados para todos os outros aplicativos sem restrições (por exemplo, o **IPC**[7]).
- Restringir dados que são compartilhados com outros aplicativos (por exemplo, implementando um provedor de conteúdo Android). Isso pode ser feito usando permissões refinadas – certifique-se de que as permissões sejam protegidas usando o nível de proteção de assinatura no Android.
- Restringir mensagens de transmissão (por exemplo, Android Broadcast Intents) para aplicativos autorizados e auditar as mensagens de transmissão do aplicativo para conteúdo sensível.
- Não permitir que teclados de terceiros sejam usados para entradas que possam conter dados sensíveis (por exemplo, credenciais, informações de cartão de crédito). **Prefira um teclado personalizado para essas entradas**[8] [9].
- Desativar a correção automática e a sugestão automática para entradas que contêm dados sensíveis.
- Desativar as funcionalidades de recortar, copiar e colar para entradas que podem conter dados sensíveis ou restringir a captura de tela para ser acessível apenas a partir deste aplicativo.
- Desativar a captura de tela para interfaces que contêm dados sensíveis. Se a plataforma não oferecer suporte a essa opção (por exemplo, iOS), notifique o usuário sobre as possíveis implicações de segurança do armazenamento de uma captura de tela em um armazenamento desprotegido.
- Desativar o plano de fundo ou usar uma tela desfocada quando o aplicativo fizer a transição para o plano de fundo em plataformas que mantêm uma captura de tela da

[7] Using Interprocess Communication https://developer.android.com/training/articles/security-tips.html#IPC.

[8] Keyboard or Keylogger?: a security analysis of third-party keyboards on Android http://seclab.skku.edu/wp-content/uploads/2015/07/mka.pdf

[9] The Samsung SwiftKey Vulnerability http://blog.trendmicro.com/trendlabs-security-intelligence/the-samsung-swiftkey-vulnerability-what-you-need-to-know-and-how-to-protect-yourself/. Acesso 01 mai. 2021.

tela visível no armazenamento local (por exemplo, iOS).

- Introduzir o mascaramento do campo de entrada para entradas que contêm dados sensíveis (por exemplo, senhas).
- Utilizar o suporte de criptografia em nível de hardware para arquivos no mais alto nível de segurança suportado. Se possível, solicitar que os arquivos do aplicativo sejam protegidos depois que o dispositivo for bloqueado.
- Se o aplicativo precisar gravar dados em um arquivo enquanto o dispositivo estiver bloqueado, usar caches temporários em vez de enfraquecer o modo de criptografia. Trocar o conteúdo do arquivo quando o dispositivo for desbloqueado e o arquivo original estiver acessível novamente.
- De preferência, usar a funcionalidade de estrutura (por exemplo, Provedor de conteúdo Android) para compartilhamento de dados em vez de usar permissões do sistema de arquivos ou um esquema de acesso personalizado em plataformas que o suportam (por exemplo, Android).
- Inspecionar mensagens de notificação personalizadas iniciadas pelo aplicativo para conteúdo sensível:
- Permitir que o usuário desative as notificações.
- Permitir que o usuário desative a exibição de conteúdo nas notificações.
- Excluir arquivos de aplicativos sensíveis dos backups de dispositivos e serviços de sincronização em nuvem. Se esta opção não estiver disponível na plataforma em uso (por exemplo, Android), exclua todo o aplicativo dos backups do dispositivo.
- Se o aplicativo permitir a seleção arbitrária de arquivos do armazenamento do dispositivo, considerar o uso de uma lista permissiva para restringir o acesso apenas aos caminhos de arquivo (absolutos) pretendidos.
- Desativar o registro do aplicativo e mensagens de depuração em versões de produção. Todas as exceções devem ser tratadas em atenção à segurança.
- No caso de o aplicativo incluir recursos de navegação na web incorporados (por exemplo, WebViews), limpar os cookies armazenados no encerramento do aplicativo ou usar o armazenamento de cookies em memória volátil.
- Não armazenar dados temporários em cache em um diretório legível por todos.

Nota: Até a conclusão deste guia, não foi localizado um procedimento padrão de exclusão segura para a memória flash (a menos que seja apagado todo o cartão). Portanto, a criptografia

de dados e o gerenciamento seguro de chaves são especialmente importantes.

2.5 Proteção dados sensíveis em trânsito

Uma das principais ameaças aos aplicativos de smartphone são os ataques baseados em rede, especialmente porque a maioria dos smartphones contém várias tecnologias de rede diferentes. Hoje, a maioria dos smartphones contém pelo menos WiFi e tecnologias de rede celular, como GPRS, UMTS, CDMA, LTE e outras. Além disso, Bluetooth e outras interfaces de rádio de curta distância, como Near Field Communication (NFC), são comumente integradas em smartphones modernos. Os dados sensíveis que passam por esses canais compartilhados podem ser **interceptados e modificados**[10].

Neste contexto seguem abaixo uma lista de tarefas que, se adotadas, trarão mais proteção e segurança à camada de rede:

- Supor sempre que a camada de rede não seja segura. Especificamente, as redes WiFi devem ser consideradas não confiáveis. Os ataques modernos à camada de rede podem derrubar a criptografia da camada de rede.
- Os aplicativos devem impor o uso de um canal seguro de ponta a ponta (como TLS) ao enviar informações sensíveis em qualquer rede (por exemplo, usando **Strict Transport Security – STS**[11]). Isso inclui passar credenciais de usuário e outros equivalentes de autenticação.
- Em caso de dados sensíveis, para reduzir o risco de ataques man-in-middle (como proxy SSL e SSL strip), uma conexão segura só deve ser estabelecida após a verificação da identidade do ponto de extremidade remoto (servidor). Isso pode ser alcançado garantindo que o TLS seja estabelecido apenas com pontos de extremidade com os certificados confiáveis na cadeia de chaves.

2.6 Incrementando segurança no tráfego de dados

- Aproveitar o suporte específico da plataforma para impor requisitos de segurança adicionais para solicitações de rede baseadas em HTTP (por exemplo, **ATS no iOS**[12] **e**

[10] SSLSNIFF: http://blog.thoughtcrime.org/sslsniff-anniversary-edition
[11] HTTP Strict Transport Security (HSTS): https://tools.ietf.org/html/rfc6797
[12] IOS https://developer.apple.com/library/ios/documentation/General/Reference/InfoPlistKeyReference/Artic les/CocoaKeys.html#//apple_ref/doc/uid/TP40009251-SW33.

desativação do tráfego de texto não criptografado no Android[13]).

- Usar algoritmos de criptografia fortes e padronizados (por exemplo, AES) e comprimentos de chave apropriados (verifique as recomendações para o algoritmo que você usa, por exemplo, para a configuração de TLS). Remova o suporte para cifras fracas.
- Aplicar **versões TLS seguras**[14]. Caso não seja possível, abortar a conexão de maneira segura.
- Usar certificados assinados por provedores de CA confiáveis. Não permitir certificados auto assinados e não desativar ou ignorar a validação da cadeia de certificados.
- Introduzir fixação de certificado (certificate pinning). Restringir os certificados confiáveis de um aplicativo a um pequeno conjunto de certificados conhecidos que são usados pelos **servidores de back-end**[15].
- Projetar a interface de uma forma que avise ao usuário se o certificado de mesmo nível não corresponder ao certificado esperado e forneça a capacidade de abortar qualquer interação posterior.
- Certificar-se de que registros adequados no servidor sejam mantidos sobre as conexões estabelecidas. No caso de vários proxies intermediários, certificar-se de que os cabeçalhos HTTP sejam analisados corretamente (por exemplo, X-Forwarded-For).
- No caso de dispositivos com acesso root ou jailbroken, considerar a integração de um contêiner seguro personalizado ou de terceiros para o canal de transmissão, uma vez que os controles de segurança da plataforma que estabelecem a conexão TLS não são confiáveis.
- Utilização de autenticação multifator
- Sempre usar estruturas com suporte de plataforma ou verificadas para estabelecer canais de comunicação seguros. Evitar usar soluções personalizadas.

2.7 Utilização de autenticação multifator

- Sempre usar estruturas com suporte de plataforma ou verificadas para estabelecer canais

[13] Android https://developer.android.com/training/articles/security-config.html. Acesso em 04 mai. 2021.
[14] Guia para a seleção, configuração e implementações de segurança da camada de transporte (TLS) http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf. Acessado em 06 mai. 2021.
[15] Certificar e atribuir chave pública https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning.

de comunicação seguros. Evitar usar soluções personalizadas.

> **Nota:** SMS e MMS devem ser evitados para enviar dados sensíveis (por exemplo, tokens de autenticação de dois fatores) para ou a partir de terminais móveis, pois **SMS e MMS podem ser interceptados**[16] [17].

3 Implementação de autenticação de usuário, autorização e gerenciamento de sessão segura

Os dispositivos móveis costumam ser compartilhados temporariamente, perdidos ou roubados. Os aplicativos móveis podem ser prejudicados por uma autenticação insegura ou controle de autorização. Indivíduos não autorizados podem obter acesso a dados ou sistemas de informação sensíveis contornando a autenticação (logins) ou reutilizando tokens ou cookies válidos.

Os aplicativos móveis devem implementar o gerenciamento de sessão segura para evitar o acesso não autorizado ao aplicativo e seus dados.

3.1 Implementando controles de autenticação e autorização

- Implementar controles de autenticação e autorização implementados no lado do servidor. Não se deve confiar nos controles de segurança do lado do cliente, considerando que os controles do aplicativo podem ser facilmente adulterados por alguém mal-intencionado.
- Considerar o uso de criptografia assimétrica para fins de autenticação e autorização. Gere e use a chave privada diretamente em um hardware seguro com suporte de plataforma, por exemplo, Trusted Execution Environment (TEE) e Secure Element (SE).
- Não revelar nomes de usuário registrados; remover qualquer impressão digital de sua existência nas mensagens de erro detalhadas.
- Usar identificadores de sessão imprevisíveis com alta entropia.

[16] SMS-based One-Time Passwords: Ataque e defesa http://www.eecs.tu-berlin.de/fileadmin/f4/TechReports/2014/tr_2014-02.pdf

[17] Segurança para Mobile Banking e pagamentos https://www.sans.org/reading-room/whitepapers/ecommerce/security-mobile-banking-payments-34062

- Certificar-se de que uma política de senha forte esteja sendo obedecida caso um mecanismo de autenticação baseado em senha seja utilizado. Considerar aplicar restrições sobre comprimento e formação da senha, reutilização de senhas de usuário antigas, uso de senhas comuns, duração da senha etc. Também pode ser útil fornecer feedback sobre a força da senha quando ela for inserida pela primeira vez. No entanto, não mantenha nenhuma representação da força da senha no armazenamento do aplicativo ou no servidor back-end, pois isso pode expor a senha em ataques de pré-imagem.

- Introduzir um mecanismo de proteção para ataques de força bruta para os controles de autenticação (por exemplo, alteração/redefinição de senha). Considere aplicar o bloqueio de conta por um período específico e perguntas estendidas sobre o usuário, notificando-o por meio de outro canal e Public Turing Tests (capcha) totalmente automatizados em caso de várias tentativas malsucedidas.

- Certificar-se de que o **gerenciamento de sessão seja tratado com segurança**[18] após a autenticação inicial, usando protocolos seguros apropriados.

- Exigir que credenciais de autenticação ou tokens sejam transmitidos com qualquer solicitação subsequente (especialmente aquelas que concedem acesso privilegiado ou modificação).

- Usar o contexto para adicionar segurança à autenticação – por exemplo, localização geográfica, localização de IP etc. Certifique-se de que **todos os dados coletados estejam em conformidade com as leis locais, a exemplo da Lei nº 13.709 (Lei Geral de Proteção de Dados Pessoais - LGPD)**[19] e outros requisitos regulatórios.

- Considerar o uso de fatores de autenticação adicionais para aplicativos que dão acesso a dados sensíveis ou interfaces, quando possível.

[18] Gerenciamento de sessão segura: segurança preventiva Voids em aplicações Web https://www.sans.org/re ading-room/whitepapers/webservers/secure-session-management-preventing-security-voids-web-applications-1594. Acesso em 02 mai. 2021.
[19] Artigos 3º (Inciso III, $1º), Art. 12 ($2º e $3º) e Art. 38 (parágrafo único) da Lei nº 13.709, Lei Geral de Proteção de Dados Pessoais http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm.

Nota: Geradores de números aleatórios geralmente produzem saída aleatória, mas previsível para uma determinada semente (por exemplo, a mesma sequência de números aleatórios é produzida para cada semente). Portanto, é importante fornecer uma semente imprevisível para o gerador de números aleatórios. O método padrão de uso de data e hora não é seguro. Ele pode ser melhorado, por exemplo, usando uma combinação de data e hora, o sensor de temperatura do telefone e os dados do sensor giroscópio (eixos x, y e z). Deve-se escolher a combinação de vários valores e o uso de algoritmos bem testados que maximizam a entropia.

- Considerar autenticação multifator para canais não seguros, tais como a Internet pública. Importa alertar que números de telefone e correios de voz podem ser sequestrados.

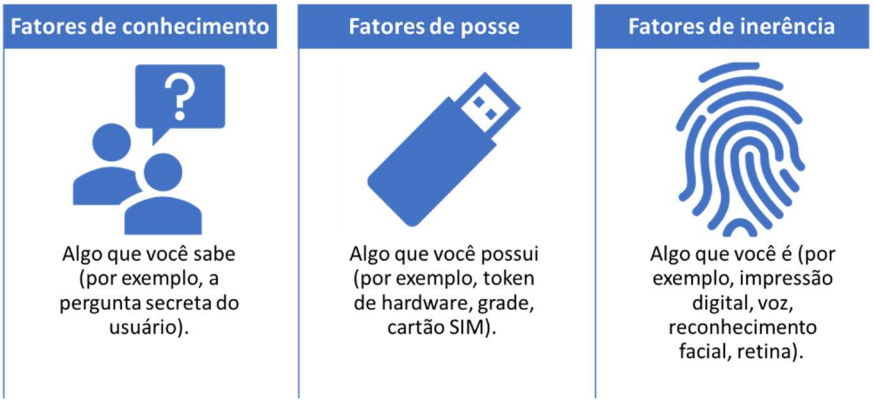


Figura 3 - Fatores de autenticação

- Usar autenticação que se vincule à identidade do usuário final (em vez de apenas à identidade do dispositivo).
- A autenticação não deve ser usada como um substituto dos controles de segurança de autorização. A autorização verifica as permissões de um usuário e pressupõe uma autenticação forte.
- Os aplicativos que oferecem suporte à autenticação do usuário devem ter uma função de logout que encerra a sessão autenticada. Após o logout, a sessão também deve ser invalidada no lado do servidor.
- Limpar todos os dados sensíveis mantidos no final da sessão. Redefina o estado do aplicativo e solicite a re-autenticação do usuário.
- Limpar todos os dados sensíveis mantidos e tente também encerrar qualquer sessão do lado do servidor após a mudança de estado do aplicativo (por exemplo, encerramento, segundo plano). Considere uma solicitação do usuário para o encerramento do aplicativo como uma solicitação de logout.

- Para aplicativos que contêm dados sensíveis, também é recomendável solicitar a re-autenticação do usuário quando o estado do aplicativo mudar para segundo plano ou verificar se o dispositivo está protegido com PIN, padrão ou senha.
- Para plataformas que suportam pilha de histórico de componente de aplicativo (por exemplo, Android), sempre limpar a pilha no encerramento da sessão ou aplicativo e a solicitação do usuário para fazer logout.
- Certificar-se de que o aplicativo seja executado com privilégio de usuário (sem privilégios administrativos) no dispositivo do usuário final, algo que não requer um dispositivo com acesso root ou desbloqueado. Verificar se ele não solicita mais autorizações de acesso aos recursos do sistema operacional e direitos no ambiente de execução do que o absolutamente necessário (princípio do menor privilégio).

4 Gerenciamento de fatores de autenticação e autorização com segurança no dispositivo

Os aplicativos móveis precisam ser projetados para proteger as credenciais do usuário e, assim, proteger os usuários, bem como a infraestrutura de back-end do aplicativo. As credenciais da conta do usuário, se roubadas, não apenas fornecem acesso não autorizado ao serviço de back-end móvel, mas potencialmente a outros serviços e contas de propriedade do usuário.

4.1 Utilização de Tokens

- Considerar o uso de tokens em vez de senhas. Tokens de autorização de longo prazo podem ser armazenados com segurança no dispositivo, de acordo com o modelo **OAuth**[20].
- Proteger os tokens em trânsito (usando TLS). Os tokens podem ser emitidos pelo serviço de back-end após verificar as credenciais do usuário inicialmente.
- Os tokens devem ser limitados por tempo ao serviço específico, bem como revogáveis (se possível do lado do servidor), minimizando assim os danos em cenários de perda. Use as versões mais recentes dos padrões de autorização (como o OAuth 2.0).
- Certificar-se de que esses tokens expirem com a maior frequência possível.

[20] Framework de Autorização OAuth 2.0 https://tools.ietf.org/html/rfc6749. Acesso em 04 mai. 2021.

4.2 Armazenamento de senhas

- No caso de senhas que necessitam ser armazenadas no dispositivo, deve-se aproveitar os mecanismos de criptografia e armazenamento de chaves fornecidos pelo sistema operacional, que são usados para armazenar senhas, equivalentes de senha e tokens de autorização. Nunca armazenar senhas em texto não criptografado. Não armazenar senhas ou IDs de sessão de longo prazo sem a criptografia apropriada.

- Aproveitar os mecanismos de armazenamento de chaves fornecidos no nível de segurança mais alto com suporte e somente quando uma senha de dispositivo for definida. Se possível, solicitar que os itens do armazenamento de chaves sejam protegidos após o dispositivo ser bloqueado e que permaneçam apenas no dispositivo atual. A título ilustrativo, sugere-se excluir esses itens dos backups e da sincronização na nuvem.

- Considerar limpar credenciais ou chaves da memória após o uso. Evitar estruturas de gerenciamento automático de memória (por exemplo, controladas pelo coletor de lixo) e objetos imutáveis para manter as chaves.

- **Apagar (zerar) imediatamente a memória que contém os dados após o uso**[21], em vez de depender de procedimentos periódicos de limpeza automática de dados. Tentar liberar os frames da Interface do Usuário (IU) imediatamente após o uso, caso as credenciais ou chaves sejam exibidas nos componentes da IU.

- Alguns **dispositivos e complementos permitem que os desenvolvedores usem um hardware seguro (por exemplo, TEE e SE)**[22] [23]- o número de dispositivos que oferecem essa funcionalidade tende a aumentar. Os desenvolvedores devem fazer uso de tais recursos para armazenar chaves, credenciais e outros dados sensíveis.

- Fornecer ao usuário móvel a capacidade de alterar senhas ou outros tokens de autenticação.

- Certificar-se de que as senhas e chaves não sejam visíveis no cache ou logs.

[21] Mutabilidade http://www.oracle.com/technetwork/java/seccodeguide-139067.html#6
[22] Hardware-Backed Keystore https://source.android.com/security/keystore/
[23] iOS Secure Enclave https://www.apple.com/business/docs/iOS_Security_Guide.pdf. Acesso 04 mai. 2021.

- Não armazenar nenhuma senha ou segredo no binário do aplicativo. Não usar um segredo compartilhado genérico para integração com o servidor de back-end (como senha embutida no código). Binários de aplicativos móveis podem ser facilmente baixados e submetidos à engenharia reversa.

5 Proteção de back-end, do servidor de plataforma e APIs

A maioria dos aplicativos móveis interage com um back-end utilizando serviços da web ou protocolos proprietários. A implementação de back-end APIs sem a segurança adequada, serviços desprotegidos e plataformas sem manutenção permitirão que, em caso de ataque, os invasores comprometam os dados no dispositivo móvel quando estiverem sendo transferidos ou **por meio do próprio aplicativo móvel**[24].

Neste capítulo, serão fornecidas apenas medidas específicas para proteger back-ends de aplicativos móveis, não substituindo a leitura de outros documentos na busca por informações complementares, tampouco a leitura do **Guia Segurança de Requisitos Mínimos de Segurança e Privacidade para APIs**[25], disponível no sítio da Secretaria de Governo Digital.

5.1 Implementando segurança em APIs

- Realizar uma verificação específica do seu código para dados sensíveis transferidos acidentalmente entre o dispositivo móvel e back-ends do servidor da web e outras interfaces externas – por exemplo, a localização ou outras informações são transferidas dentro dos metadados do arquivo.
- Testar periodicamente as vulnerabilidades de todos os serviços de back-end (serviços da web) para aplicativos móveis. Por exemplo, usando ferramentas de análise de código estático e ferramentas de difusão para testar e encontrar falhas de segurança. Executar o teste de stress e de caso de uso.
- **Desativar a publicação de metadados**[26] (por exemplo, metadados para documentos WSDL e para objetos derivados de WSDL), a fim de evitar a divulgação não intencional de metadados de serviço potencialmente sensíveis.

[24] OWASP Web Services https://www.owasp.org/index.php/Web_Services. Acessado em 06 mai. 2021.
[25] https://www.gov.br/governodigital/pt-br/seguranca-e-protecao-de-dados/guias/guia_seguranca_apis.pdf
[26] Publicação de metadados padrão em .NET Framework https://msdn.microsoft.com/en-us/library/ms751498(v=vs.110).aspx

- Certificar-se de que a plataforma de back-end (servidor) esteja executando com uma configuração reforçada com os patches de segurança mais recentes aplicados ao sistema operacional, servidor da web e outros componentes do aplicativo.
- Garantir que registros adequados sejam mantidos no back-end para **detectar e responder a incidentes**[27] e realizar análises forenses (nos termos da Lei 13.709, Lei Geral de Proteção de Dados Pessoais - LGPD). Recomendamos a leitura do **Guia de Resposta a Incidentes de Segurança disponível**[28] no sítio da Secretaria de Governo Digital.
- **Proteger o back-end de log injections**[29] iniciadas pelo cliente que podem corromper ou forjar o histórico de eventos.
- Implementar controle de banda (QoS) por usuário/IP (se a identificação do usuário estiver disponível) para reduzir o risco de ataque de negação de serviço (DoS).
- Testar as vulnerabilidades DoS em que o servidor pode ficar sobrecarregado por certas chamadas de aplicativos que consomem muitos recursos.

6 Integração segura de dados com código de terceiros

A utilização de código de terceiros pode trazer consigo riscos de segurança e privacidade. Eles podem usar o acesso do aplicativo aos dados do usuário e vazá-los propositalmente ou por acidente, como também podem introduzir vulnerabilidades de segurança em um aplicativo considerado seguro. Os desenvolvedores de aplicativos devem realizar uma análise mais profunda do código de terceiros que venham a incluir em seu aplicativo.

[27] Artigo 48 da Lei nº 13.709, Lei Geral de Proteção de Dados Pessoais http://www.planalto.gov.br/ccivil_03/_ Ato2015-2018/20 18/Lei/L13709.htm.
[28] https://www.gov.br/governodigital/pt-br/seguranca-e-protecao-de-dados/guias/guia_resposta_incidentes.pdf
[29] OWASP Log Injection https://www.owasp.org/index.php/Log_Injection. Acessado em 06 mai. 2021.

6.1 Utilizando código de terceiros

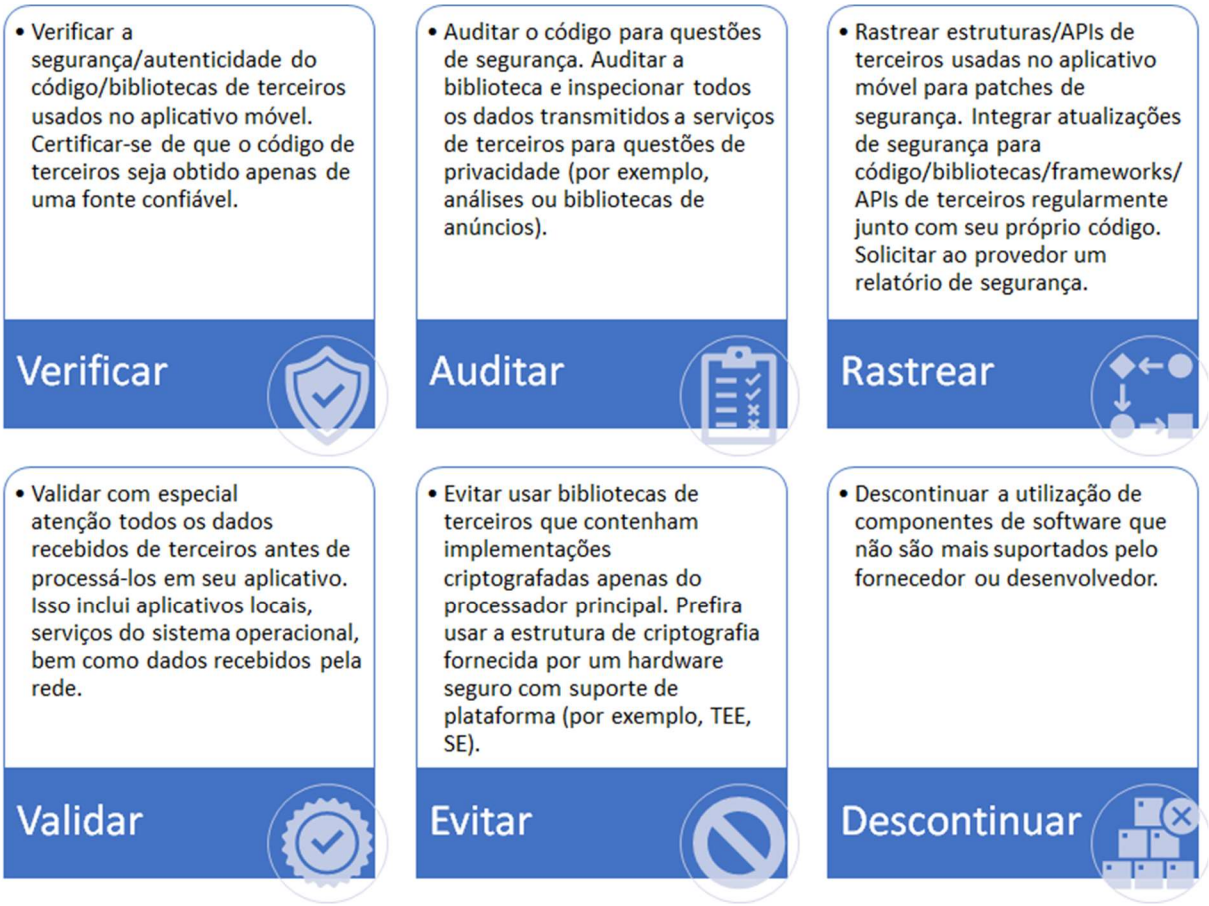


Figura 4 – Recomendações com de códigos de terceiros

7 Proteção de privacidade e consentimento

Os aplicativos móveis geralmente armazenam e operam com informações pessoais. Portanto, eles **precisam ser projetados para evitar a divulgação não intencional de informações pessoais ou privadas, a exemplo da Matriz PET**[30].

Para a garantia da transparência, recomenda-se que os desenvolvedores tenham atenção especial para que **o usuário seja informado a respeito de qualquer operação de tratamento e uso compartilhado que ocorra no dispositivo pelo aplicativo**[31] [32]. Tal informação deverá ser fornecida de maneira transparente, detalhada, de fácil entendimento e solicitando consentimento do usuário caso necessário.

[30] Matriz de controle PET - avaliando ferramentas de privacidade on-line e mobile https://www.enisa.europa.eu/publications/pets-controls-matrix/pets-controls-matrix-a-systematic-approach-for-assessing-online-and-mobile-privacy-tools. Acessado em 06 mai. 2021.
[31] Artigo 7º, Inciso X, Parágrafo 5º da Lei nº 13.709, Lei Geral de Proteção de Dados Pessoais.
[32] Artigo 11º, Incisos I e II, Lei Geral de Proteção de Dados Pessoais http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm.

Mais detalhes quanto às particularidades sobre o consentimento que devem ser observadas no desenvolvimento de aplicativos são abordados a seguir.

7.1 Coletando dados pessoais

- Verificar se o aplicativo está coletando dados pessoais. Pode não ser sempre óbvio, um exemplo de ação é analisar se estão sendo utilizados identificadores únicos persistentes vinculados a armazenamentos centrais de dados contendo informações pessoais.
- Cumprir a política de privacidade e proteção de dados pessoais do órgão estabelecido para o serviço/aplicativo em questão, cobrindo o uso de dados pessoais e disponibilizar ao usuário, especialmente antes do pedido de autorização para coleta e compartilhamento de dados pelo aplicativo.
- Considerar aproveitar os recursos integrados para exigir acesso aos sensores e dados do dispositivo – por exemplo, acesso a GPS, câmera etc. Forneça uma explicação clara sobre o motivo pelo qual o acesso é necessário.
- Minimizar o acesso aos dados do sensor sempre que possível – por exemplo, não coletar dados de geolocalização automaticamente se tal recurso não for estritamente necessário.
- Reduzir o armazenamento detalhado dos dados e torná-los anônimos no dispositivo móvel ao invés de remotamente – por exemplo, remover metadados de imagem.
- Reduzir o período de retenção de dados no celular ou remotamente ao mínimo necessário para fornecer o serviço. Excluir os dados imediatamente após o período de retenção expirar. Excluir dados de todos os locais, especialmente servidores remotos, onde os dados podem ser armazenados.
- Usar tecnologias que aumentem a privacidade, que suportem a **minimização de dados, anonimização e segurança de dados pessoais**[33] [34].

[33] Artigo 5º, Inciso XI da Lei nº 13.709, Lei Geral de Proteção de Dados Pessoais http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm. Acesso 06 mai. 2021.

[34] Mais informações em https://www.enisa.europa.eu/topics/data-protection/privacy-enhancing-technologies.

- As configurações do aplicativo devem fornecer privacidade e segurança a exemplo dos padrões Privacy by Default e Security by Default, respectivamente. O primeiro trata da ideia que o serviço traz para o usuário o padrão todas as medidas de privacidade de dados que foram concebidas durante o desenvolvimento. O segundo, por sua vez, recomenda as configurações padrão do aplicativo como as mais seguras possíveis, sem a necessidade de ajustes por parte do usuário.

- Exigir o **consentimento do usuário**[35] antes de fornecer os dados a terceiros. Forneça um aviso claro sobre o aplicativo de compartilhamento de dados com terceiros. Nunca forneça dados de localização precisos para aplicativos de terceiros, nem dados armazenados em contêineres seguros do aplicativo.

7.2 Obtendo consentimento

Poderá haver cenários que, para garantia da conformidade com a regulamentação, seja necessário obter o consentimento do usuário antes da utilização de dados pessoais. Convém que, na fase de projeto, seja avaliado se o **consentimento**[36] será utilizado e, caso seja essa a opção, os requisitos para se garantir a legalidade e a operacionalização do consentimento sejam atendidos pelo aplicativo.

As recomendações a seguir são relativas exclusivamente às hipóteses em que o consentimento seja utilizado como meio, ou base legal, para a obtenção de uma determinada funcionalidade do aplicativo.

Transparência:

Ao obter o consentimento, notifique explicitamente o usuário com informações específicas relativas aos dados que foram consentidos. O aplicativo deve oferecer meios para que o usuário possa revisar suas preferências de consentimento e revogá-lo, caso deseje. Exemplo de informações que podem constar nas notificações sobre consentimento são:

- Quais exatamente são os dados pessoais que serão usados;
- Qual é a finalidade do processamento;

[35] Inciso XII do Artigo 5º da Lei nº 13.709, Lei Geral de Proteção de Dados Pessoais http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm
[36] Artigo 7º da Lei nº 13.709, Lei Geral de Proteção de Dados Pessoais.

- Quem são os destinatários dos dados (controladores, controladores conjuntos, operadores, terceiros etc.);
- Onde os dados são armazenados e por quanto tempo.

Momento oportuno:

O consentimento pode ser obtido de 3 maneiras principais:

- No momento da instalação;
- Em tempo de execução quando os dados são enviados;
- Por meio de mecanismos "opt-in", nos quais um usuário deve ativar explicitamente uma configuração.

Impactos na funcionalidade:

Caso o usuário não conceda consentimento a todos os dados solicitados, ele deve ser informado sobre as possíveis limitações da funcionalidade do aplicativo.

Revogação do consentimento:

Deve ser possível para o usuário retirar o consentimento de forma facilitada a qualquer momento no aplicativo. Notifique o usuário sobre como o comportamento e a funcionalidade do aplicativo poderão ser afetados caso o consentimento seja retirado.

Manutenção de registros:

Manter registros do consentimento dado pelo usuário para o tratamento de diferentes tipos de dados pessoais. Recomenda-se que os registros contenham o momento em que o consentimento foi obtido (data/hora), em qual etapa do processo ele foi fornecido, para qual finalidade e para quais elementos de dados.

Especial atenção deve ser concedida à interface do usuário e/ou à experiência do usuário. Em linhas gerais, com base nas considerações do art. 5º da LGPD em seu inciso XII, recomenda-se o atendimento aos seguintes tópicos:

Manifestação livre:

A funcionalidade principal do aplicativo não poderá decair na hipótese de que o usuário opte por não consentir a um propósito que seja acessório. A exceção será relativa a

uma determinada funcionalidade que dependa irrestritamente do dado pessoal vinculado ao consentimento.

Manifestação informada: O usuário deve ter informações claras em relação ao que ele está consentindo e para qual finalidade. Essa informação pode ser fornecida, por exemplo, na forma de um banner, para que o usuário tenha clareza de que é de seu livre arbítrio autorizar ou não que tais informações sejam processadas, caso ele queira alguma funcionalidade que seja suplementar ao serviço oferecido.

Exemplo:

Considere um aplicativo de funcionários que permita aos usuários funcionalidades de baixar o contracheque e enviar dados para solicitação de seguro saúde para os dependentes.

É obrigação da empresa disponibilizar o contracheque. Portanto, esse seria um propósito principal. Todavia, suponha que o aplicativo peça consentimento para enviar dados à seguradora: na hipótese de que o usuário não forneça consentimento, ele ficará impedido de ter o referido seguro. Porém, não seria adequado por conta disso limitar ao usuário a possibilidade de baixar seu contracheque, sendo este último um propósito principal não relacionado ao consentimento.

Manifestação informada:

O usuário deve ter informações claras em relação ao que ele está consentindo e para qual finalidade. Essa informação pode ser fornecida, por exemplo, na forma de um banner, para que o usuário tenha clareza de que é de seu livre arbítrio autorizar ou não que tais informações sejam processadas, caso ele queira alguma funcionalidade que seja suplementar ao serviço oferecido. Pontos relevantes desse tópico são:

- **Clareza:** Deve ser claro ao usuário o que ele está consentindo, os benefícios que ele poderá ter caso concorde, bem como eventuais restrições que ele poderá observar na hipótese de não consentir.
- **Finalidade:** Deverá estar claro o motivo ou a finalidade, que tais dados precisarão ser processados e os eventuais riscos à sua privacidade na hipótese de concordância.

- **Adequabilidade:** A linguagem deve ser clara, simples e adequada ao público ao qual se destina. A título exemplificativo, um aplicativo desenvolvido para o público infantil deve possibilitar o entendimento das crianças e considerar a eventual necessidade de confirmação que tal consentimento foi fornecido pelo responsável do menor.

Manifestação inequívoca:

O consentimento deve advir de uma ação do usuário, quer seja pela seleção de uma caixa de opção (check-box), quer seja do envio de um e-mail ou qualquer forma de manifestação que parta de uma ação positiva do usuário.

Portanto, o aplicativo deve oferecer diversos mecanismos que possibilitem a adequação e a gestão do consentimento, tanto pelo usuário quanto pelo órgão.

7.3 Auditoria na coleta de dados do usuário

- Auditar mecanismos de comunicação para verificar vazamentos indesejados (por exemplo, metadados de imagem).
- Verificar se a coleta de dados do dispositivo não é excessiva em relação à autorização ou ao consentimento feito pelo usuário – por exemplo, coletar mais tipos de dados do que o necessário - APP-native + WebKit HTML. Além de analisar se a coleta de dados segue os princípios estabelecidos no art. 6º da LGPD, em especial os incisos I e III (finalidade e necessidade).

8 Proteção para recursos de pagamento

Os aplicativos de smartphone fornecem acesso programático a recursos de pagamentos em telefones celulares, como chamadas, SMS, dados em roaming, pagamentos NFC e sistemas de pagamento de terceiros. Os aplicativos que integram esses serviços devem ter um cuidado especial para evitar abusos. O processo de desenvolvimento deve considerar o impacto financeiro das vulnerabilidades nos aplicativos criados. Além disso, os aplicativos que implementam o pagamento interno para oferecer serviços ao usuário devem proteger seu código de pagamento contra abusos.

8.1 Registro dos recursos de pagamentos

- Manter registros de acesso a recursos de pagamentos em formato não-repúdio (non-repudiable) – por exemplo, um recibo assinado enviado a um back-end de servidor confiável, com o consentimento do usuário. Disponibilizar ao usuário final para monitoramento. Os logs devem ser protegidos de partes não autorizadas.

8.2 Precauções com recursos de pagamento

- Avisar ao usuário e obter consentimento para quaisquer implicações de custo para o comportamento do aplicativo.
- Seguir as diretrizes do fornecedor do sistema operacional/dispositivo para implementar o pagamento no aplicativo.
- **Implementar a validação de recibos de pagamento no servidor de back-end**[37], e não no dispositivo. Atenção especial ao integrar a aceitação de pagamento de uma carteira de terceiros (carteira não integrada ao sistema operacional móvel).
- Verificar se há padrões de uso anômalos no uso de recursos de pagamentos e acionar a re-autenticação – por exemplo, quando ocorrerem mudanças significativas no local, alterações de idioma do usuário, uso de serviço pago por pagamento significativamente mais alto.
- Considerar um modelo de allow-list por padrão para endereçamento de recursos pagos – por exemplo, contatos do catálogo de endereços apenas, a menos que especificamente autorizado para chamadas telefônicas.

8.3 Controle de acesso

Recomenda-se a utilização de recursos com diferentes níveis de controle de acesso, dependendo da versão do sistema operacional.

Para evitar abusos, especial atenção deve ser dada à implementação do controle de acesso em sistemas de informação que utilizem recursos de pagamento. Tal ação deve considerar possíveis impactos que versões mais antigas ou mais recentes do sistema operacional e/ou estrutura possam sustentar no nível de controle de acesso e na segurança esperada.

[37] VirtualSwindle: Ataque automatizado contra o In-App Billing no SO Android http://mulliner.org/collin/publications/asia226-mulliner.pdf. Acessado em 06 mai. 2021.

9 Distribuição segura de software

A segurança geral do software em dispositivos móveis é imposta por assinatura de código e atualizações de segurança rápidas. O uso de práticas seguras para distribuição de software é fundamental para a segurança geral do aplicativo e é fundamental para mitigar todos os riscos descritos neste guia.

É importante estar atento à administração de aplicativos e gerenciamento de identidade e acesso (IAM) nas plataformas e console de aplicativos (Appstore). Não é raro encontrar problemas de gerenciamento de identidades, tais como usuários compartilhados ou com mais privilégios que o necessário e aplicativos esquecidos.

Recomenda-se fortemente atribuir o gerenciamento de identidade e acesso a uma equipe que seja responsável pela atribuição de privilégios e reduzir para a menor quantidade possível os usuários com funções de:

- Administrador (uma vez que podem adicionar ou excluir outros usuários dentro do console associados ao aplicativo que é administrado);
- Publicador (já que possibilita a publicação ou republicação de aplicativos); e
- Suporte ao cliente (pois permite responder aos comentários do usuário que utiliza o aplicativo).

Nota: No contexto administrativo dos órgãos, considerando as características e a sua relevância, entende-se como razoável atribuir a permissão de administrador para, pelo menos, 2 (duas) pessoas da equipe técnica do órgão ou entidade – uma atuando como titular e a outra como suplente ou substituta nas ausências oficiais do titular.

Publicação do aplicativo

- Os aplicativos devem ser projetados e provisionados para permitir atualizações de patches de segurança, levando em consideração os requisitos para aprovação pelas lojas de aplicativos e o atraso extra que isso pode implicar.
- As lojas de aplicativos oficiais monitoram aplicativos em busca de código inseguro e, no caso de um incidente, são capazes de remover aplicativos remotamente em curto prazo. Portanto, distribuir aplicativos por meio de lojas de aplicativos oficiais, fornece uma rede de segurança em caso de vulnerabilidades graves em seu aplicativo.

- Fornecer canais de feedback para que os usuários relatem problemas de segurança com aplicativos, como um endereço **security@email, por exemplo**[38].

Nota: Para mais informações acerca dos canais de comunicação, recomenda-se a leitura a partir do item 2.8 do Guia de Resposta a Incidentes de Segurança.

- Se uma loja de aplicativos corporativa for usada, proteja a chave de assinatura do aplicativo com o máximo cuidado – por exemplo, use um HSM, máquina com air-gap etc.
- Em última instância, as atualizações de segurança podem ser enviadas fora da Appstore, no entanto deve-se usar uma conexão criptografada e seu conteúdo deve ser verificado – por exemplo, com uma ferramenta anti-malware – antes de aplicar a atualização.
- Os recursos usados por aplicativos atualizados fora do mecanismo normal da loja de aplicativos devem ser assinados. Os aplicativos devem verificar a assinatura antes de aceitar o recurso atualizado.
- Não implantar aplicativos com certificados de assinatura *ad-hoc* usados para desenvolvimento e teste.
- Não gerar um aplicativo para vários ambientes. O aplicativo de produção não deve conter chamadas de log, URLs de desenvolvedor, métodos de teste e configurações do ambiente de desenvolvimento ou teste.
- Comprovar, por meio de ferramentas e/ou metodologias aceitáveis, por exemplo **MobSF – Mobile Secure Framework**[39] (https://github.com/MobSF/Mobile-Security-Framework-MobSF), que o aplicativo de lançamento se encontra em um patamar de risco considerado baixo.

[38] Recomendações para caixas de correios https://www.ietf.org/rfc/rfc2142.txt. Acessado em 06 mai. 2021.
[39] Mobile Secure Framework https://github.com/MobSF/Mobile-Security-Framework-MobSF. Acesso 24 nov. de 2021.

10 Interpretação do tempo de execução de código

A interpretação do código em tempo de execução e o tratamento descuidado do fluxo de informações podem dar uma oportunidade para que partes não confiáveis forneçam informações não verificadas que sejam interpretadas como código ou vazem informações sensíveis. Isso dá uma oportunidade para o malware contornar os Walled Garden Control fornecidos por lojas de aplicativos. Pode levar a ataques de injeção que levam ao vazamento de dados, vigilância e spyware.

A falta de controle no fluxo e manipulação de informações pode levar ao vazamento de dados.

Observe que nem sempre é óbvio que seu código contenha um interpretador. Procure por quaisquer recursos acessíveis por meio de dados de entrada do usuário e uso de APIs de terceiros que possam interpretar a entrada do usuário - como interpretadores de JavaScript, por exemplo.

10.1 Orientações para segurança de interpretadores de códigos

- Filtrar os dados do usuário passados para os interpretadores.
- Definir uma sintaxe de escape abrangente, conforme apropriado.
- Não revelar informações sensíveis, como nomes de usuário, dados pessoais e outros, por meio de mensagens de erro.
- Negar acesso direto do código interpretado aos dados do usuário e armazenamento criptografado.
- Retirar as funcionalidades não utilizadas dos intérpretes.
- Limitar o tamanho dos dados de entrada passados para os interpretadores.

11 Verificação da integridade do dispositivo e do aplicativo

Dispositivos e/ou aplicativos modificados prejudicam os controles de segurança e privacidade implementados no aplicativo móvel. A modificação do dispositivo pode ser feita por meio de enraizamento/desbloqueio ou instalando uma imagem de sistema operacional personalizada. Não se pode confiar que os aplicativos modificados se comportem da maneira desejada pelo desenvolvedor. O mesmo conta para dispositivos modificados. As plataformas atuais de smartphone suportam recursos de verificação de integridade de

dispositivo e/ou aplicativo que devem ser aproveitados.

11.1 Integridade do dispositivo

- Verifique a integridade do dispositivo/plataforma para garantir que o dispositivo não seja modificado. Prefira usar os serviços da plataforma, se disponíveis – por exemplo, **Android SafetyNet attestation**[40]. Apenas implemente ou use detecção de root/jailbreak de terceiros se a plataforma não oferecer uma solução integrada.
- Verificar a integridade do aplicativo e se o aplicativo e seus recursos não foram modificados.
- Usar o serviço da plataforma (por exemplo, **Android SafetyNet attestation, iOS App Store receipt**[41]).
- Executar verificações de integridade de código na memória para proteger contra modificação de código e/ou interceptação de tempo de execução.

11.2 Desativando os recursos do desenvolvedor

- Desativar a depuração nas configurações do aplicativo.
- Verificar se o dispositivo está no modo de desenvolvedor se compatível com a plataforma (por exemplo, Android).
- Verificar se o depurador está conectado e/ou se o processo está sendo rastreado. Em plataformas com código gerenciado, verificar se há depuradores de código gerenciado e nativo.
- Tornar a engenharia reversa mais difícil, ofuscando o código e criptografando os dados (por exemplo, strings) para proteger ainda mais a lógica do aplicativo.

[40] Verificando a compatibilidade do dispositivo com SafetyNet https://developer.android.com/training/safetynet/index.html.
[41] IOS Receipt Validation https://developer.apple.com/library/ios/releasenotes/General/ValidateAppStoreReceipt/Introduction.html. Acessado em 06 mai. 2021.

12 Proteção do aplicativo contra injeções do lado do usuário

Os aplicativos móveis apresentam maiores oportunidades de injeções do lado do cliente, uma vez que interagem constantemente com sensores, outros aplicativos instalados e serviços de terceiros. As falhas de aplicativos móveis existentes podem ser exploradas de maneira semelhante às vulnerabilidades em aplicativos de software tradicionais. Os invasores podem forçar o aplicativo a usar dados especialmente criados que modificarão o fluxo lógico do aplicativo e levarão a desvios de controle de acesso ou ataques de divulgação de informações.

12.1 Reduzindo o risco de ataques por injeção de código

- No caso de o aplicativo incluir recursos de navegação na web incorporados (por exemplo, WebViews), restringir o acesso a domínios de terceiros que não estejam em conformidade com os padrões de segurança exigidos, desativar quaisquer funcionalidades não utilizadas da plataforma suportada, como plug-ins, acessibilidade de arquivo local acessibilidade do provedor de conteúdo (URL de conteúdo) e suporte de execução de código dinâmico (por exemplo, JavaScript).
- Evitar usar interfaces da web de tela inteira, pois elas podem ser utilizadas por invasores para criar telas de aplicativos falsas.
- Evitar usar chamadas de API que fornecem ponte de código dinâmico (por exemplo, JavaScript) com código nativo (por exemplo, Objective-C), pois uma **injeção no código dinâmico levará à execução do código nativo**[42].
- No caso em que o aplicativo usa código JavaScript em execução no contexto de um URL de esquema de arquivo, é recomendável desabilitar todos os atributos não utilizados da plataforma, como acessar conteúdo de outra URL de esquema de arquivo e conteúdo de qualquer origem.

[42] Barrando ataques: WebView Exploitation https://www.usenix.org/system/files/conference/leet13/leet13-paper_neugschwandtner.pdf. Acessado em 06 mai. 2021.

- Evitar eventos de interação quando o aplicativo é obscurecido por outra interface na camada de apresentação, a fim de **mitigar ataques de tapjacking**[43]. Ao desativar os eventos de interação do aplicativo, a possibilidade de um usuário interagir com uma visualização oculta é eliminada.

- Caso o aplicativo solicite **permissões customizadas**[44] e plataformas mais antigas sejam compatíveis (por exemplo, anteriores ao Android 5.0), sempre verificar na primeira execução do aplicativo se nenhum outro aplicativo solicitou as mesmas permissões anteriormente.

- Sempre siga a infraestrutura de registro de nome de domínio (DNS) para declarar uma permissão personalizada, a fim de evitar colisões com outros aplicativos.

- Restringir quais aplicativos podem fazer com que um componente do aplicativo (por exemplo, atividade do Android) seja iniciado ou seja capaz de interagir com ele (por exemplo, serviço e provedor de conteúdo do Android). Isso pode ser feito usando permissões restritas.

- Restringir os aplicativos de terceiros cujas mensagens de difusão serão aceitas.

- No caso de o aplicativo utilizar um gerenciador de download fornecido pela plataforma, sempre verifique se as notificações do gerente recebidas estão relacionadas aos downloads iniciados pelo aplicativo.

- Sempre verifique os downloads de código dinâmico e as atualizações do aplicativo no lado do cliente. Qualquer recurso que está sendo recuperado de um serviço externo (por exemplo, arquivos compactados, arquivos APK) deve ser validado quanto à integridade e ao certificado de assinatura.

- Sempre validar as respostas do servidor ao usar APIs de back-end. Apresente um modelo de lista de permissões para respostas aceitas.

[43] Tapjacking http://blog.trendmicro.com/trendlabs-security-intelligence/tapjacking-an-untapped-threat-in-android/
[44] O problema de permissão customizada https://github.com/commonsguy/cwac-security/blob/master/PERMS.md. Acessado em 06 mai. 2021.

- Mitigar injeções de SQL, inclusão de arquivo local, injeções de JavaScript, injeções de XML. Ao lidar com consultas dinâmicas (por exemplo, consultas SQL com entradas não confiáveis) ou provedores de conteúdo, certifique-se de usar consultas parametrizadas. Sempre valide as entradas fornecidas pelo usuário que serão usadas para fins de acesso a arquivos ou como parte de uma execução dinâmica de código. Use uma estrutura verificada para operações XML.

- Proteger contra corrupções de memória em aplicativos que são desenvolvidos usando uma linguagem de programação que suporta gerenciamento de memória explícito (por exemplo, Objective-C, C, C++). Executar análises estáticas para vulnerabilidades de gerenciamento de memória no processo de desenvolvimento.

- Não usar dados em cache inseguros em conexões HTTP e em recursos de navegação na web incorporados (por exemplo, WebViews). Os caches geralmente estão localizados no sistema de arquivos do dispositivo. Muitas plataformas permitem que os aplicativos coloquem esses dados em cache em locais inseguros (por exemplo, sdcard no Android) nos quais eles podem ser facilmente adulterados.

- Em plataformas que oferecem suporte a aplicativos personalizados com permissões de acessibilidade (por exemplo, Android), evitar que elementos sensíveis da interface do usuário sejam acessados por aplicativos de acessibilidade.

- Evitar preencher visualizações da web carregadas do esquema URI do arquivo com a entrada DOM fornecida pelo usuário.

13 Garantindo o uso correto de sensores biométricos e hardware seguro

Os sensores biométricos tornam os sistemas de autenticação mais fáceis e rápidos de usar. No entanto, as políticas de autenticação e acessibilidade devem ser aplicadas pelo hardware seguro para serem protegidas contra qualquer coisa, incluindo o comprometimento do kernel.

13.1 Utilizando recursos biométricos com segurança

- Sempre verificar se há um sensor biométrico (por exemplo, leitor de impressão digital) presente e disponível no dispositivo antes de usar a API para fins de autenticação. Caso o sensor não esteja disponível, um controle de autenticação alternativo deve ser fornecido.

- Sempre verificar se o sensor biométrico e a política de autenticação de hardware seguro da plataforma em uso estão em conformidade com a política de autenticação do aplicativo (senha exigida após inicialização a frio, expiração de autenticação do sensor biométrico, adicionar uma impressão digital requer PIN/senha/autenticação biométrica, requisito para sensor biométrico sendo emparelhado individualmente com hardware seguro).
- Certificar-se de que haja dados registrados usando o sensor biométrico – por exemplo, as impressões digitais do usuário e/ou a íris do usuário estão registradas – antes de usar a API para fins de autenticação.
- Certificar-se de que os dados biométricos registrados não foram alterados desde a ativação do controle de autenticação usando o sensor biométrico – por exemplo, outro usuário adicionou uma nova amostra de impressão digital/íris.
- O aplicativo não deve usar o sensor biométrico apenas para verificar a presença do usuário (por exemplo, iOS LocalAuthentication). Este controle pode ser facilmente contornado usando dynamic hooking/static patching. Em vez disso, o aplicativo deve usar o sensor biométrico para acessar as chaves armazenadas usando um keystore/keychain com suporte de hardware e protegidas com listas de controle de acesso (ACL).
- Certificar-se de que o material da chave esteja vinculado ao hardware seguro (por exemplo, TEE, SE) em plataformas que sejam opcionais (por exemplo, Android). Quando esse recurso é habilitado para uma chave, o material da chave nunca é exposto fora do hardware seguro.
- Para chaves cujo material está dentro de um hardware seguro (por exemplo, TEE, SE), certificar-se de que as autorizações criptográficas e de autenticação do usuário também sejam aplicadas por hardware seguro, em plataformas que sejam opcionais (por exemplo, Android). O controle de autenticação (por exemplo, usando verificações biométricas) e a descriptografia da chave devem ser realizados atomicamente em um TEE ou em um chip com canal seguro para o TEE.

- O aplicativo deve evitar o uso de autorizações de intervalo de validade temporal, uma vez que é improvável que sejam aplicadas pelo hardware seguro porque normalmente não tem um relógio de tempo real seguro independente.
- Verificar se a política de autenticação do aplicativo está em conformidade com a possibilidade de que diferentes pessoas possam se inscrever para autenticação biométrica no mesmo dispositivo. Como resultado, a autenticação biométrica bem-sucedida pode ser possível para diferentes usuários de dispositivos.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT NBR ISO/IEC 27001:2013. Tecnologia da Informação - Técnicas de Segurança

ABNT NBR ISO/IEC 27005:2011. Tecnologia da informação – Técnicas de segurança – Gestão de riscosde segurança da informação.

ABNT NBR ISO/IEC 27701:2019. Técnicas de segurança — Extensão da ABNT NBR ISO/IEC 27001 ABNT NBR ISO/IEC 27002:2013 para gestão da privacidade da informação — Requisitos e diretrizes.

ABNT NBR ISO/IEC 29134:2017. Information technology – Security techniques – Guidelines for privacy impact assessment.

ABNT NBR ISO/IEC 29151:2017. Information technology – Security techniques – Code of practice for personally identifiable information protection.

ABNT NBR ISO/IEC 31000:2018. Gestão de riscos - Diretrizes.

BRASIL. Presidência da República. Casa Civil. Subchefia para Assuntos Jurídicos. **Lei nº 13.709**, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais.

Disponível em: < http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm >. Acesso em: junho 2021.

AGÊNCIA DA UNIÃO EUROPÉIA DE SEGRANÇA DA INFORMAÇÃO - ENISA. **Smartphone Secure Development Guidelines**. Maio 2021.

Disponível em: < https://www.enisa.europa.eu/publications/smartphone-secure-development-guidelines-2016/at_download/fullReport >. Acesso em maio 2021.

M. Colesky, J.-H. Hoepman e C. Hillen, "A Critical Analysis of Privacy Design Strategies," in International Workshop on Privacy Engineering - IWPE'16, San Jose, CA, EUA, 2016.

COMITÊ CENTRAL DE GOVERNANÇA DE DADOS - CCGD. **Guia de Boas Práticas LGPD**. junho 2021. Disponível em: < https://www.gov.br/governodigital/pt-br/governanca-de-dados/guia-de-boas-_praticas-lei-geral-de-protecao-de-dados-lgpd >. Acesso em: junho 2021.

J. H. Hoepman., "Privacy Design Strategies," in IFIP TC11 29th Int. Conf. sobre Segurança da Informação (IFIP SEC 2014), 2014.

LEGISLAÇÃO GSI/PR

Disponível em: < https://www.gov.br/gsi/pt-br/assuntos/dsi/legislacao > Acesso em: junho 2021

OPEN WEB APPLICATION SECURITY PROJECT - OWASP. **Mobile Security Project Archive**. Maio 2021.

Disponível em: < https://wiki.owasp.org/index.php/Mobile_Security_Project_Archive#tab=Top_10_Mobile_Controls >.

Acesso em: maio 2021.

PORTARIA Nº 93, DE 26 DE SETEMBRO DE 2019 - Glossário de Segurança da Informação INSTRUÇÃO NORMATIVA Nº 4, DE 26 DE MARÇO DE 2020 - Segurança Cibernética em redes 5G DECRETO Nº 10.222, DE 5 DE FEVEREIRO DE 2020 - Estratégia Nacional de Segurança Cibernética.