セキュアコーディングガイド最新版の紹介

JSSECセキュアコーディング ワーキング グループ

Jun.Ogiso@sony.com

About me

name: 小木曽　純
role:　JSSEC セキュアコーディング ワーキンググループ リーダー
org:　ソニーデジタルネットワークアプリケーションズ（株）
job:　Secure Coding Checker

ガイドの紹介

安全なAndroid™アプリの作り方

業界標準

2012年から継続的に改訂

お好きな形式で読めます

英語版も公開

HTML版

PDF版

https://developer.android.com/about/versions/pie/

ガイド変更点

Autofillフレームワーク対策
アプリ署名検証API
サーバー証明書検証
暗号化通信
指紋認証機能

Safe Browsing
共有メモリ

Autofill framework

Android 8.0 (API Level 26)

入力文字をAutofill Serviceが保存

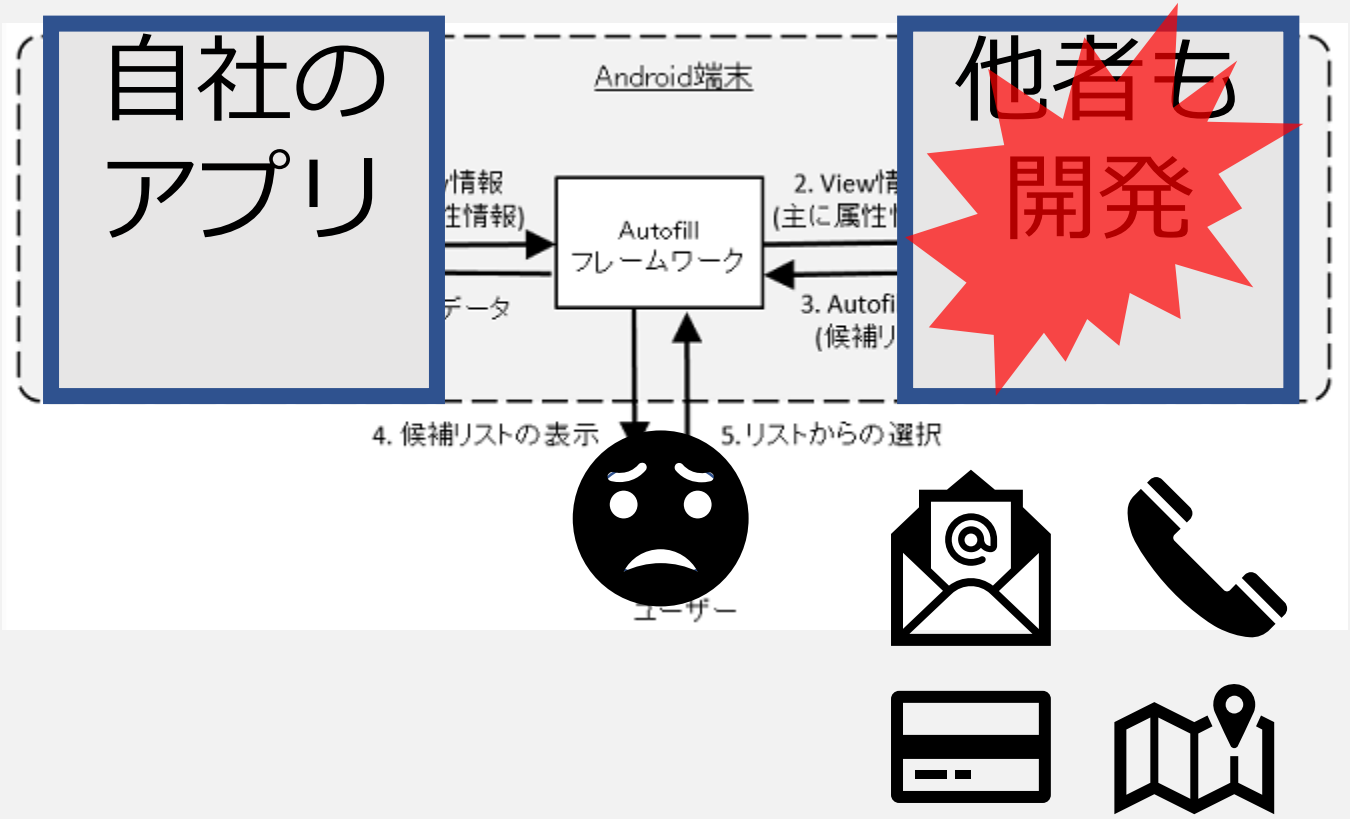
次回からAutofill Serviceが自動入力

住所、パスワード、クレジットカード番号もAutofill Serviceに渡る

Android 8.0/8.1のAutofill

Autofill Serviceは
　　自由に作れる
　　自由に配れる

導入・利用は
　　ユーザーの自由
　　ユーザーの判断



Android 8.0/8.1のAutofill対策

アプリがAutofillを「使わせない」

ViewごとにAutofillの対象外指定

長押しフローティング ツールバー削除

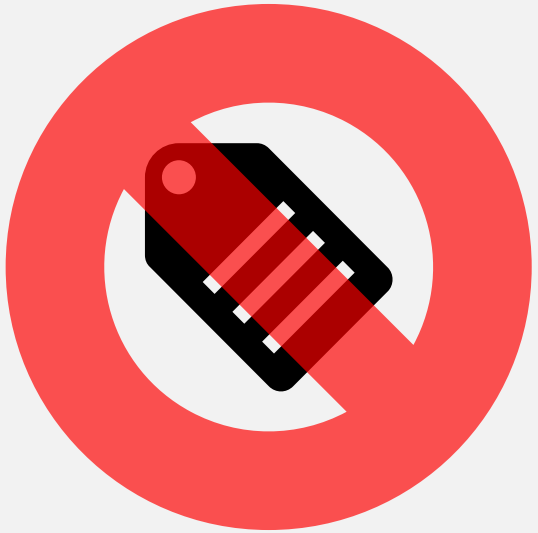
Android 9.0のAutofill強化ポイント

AutofillManagerクラス

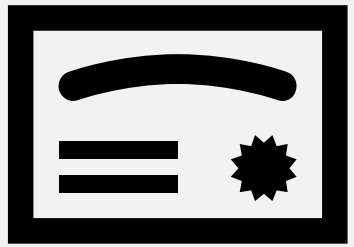
getAutofillServiceComponentName()

Autofill Serviceのパッケージ名がわかる

Android 9.0のAutofill対策

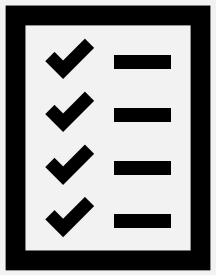


パッケージ名で判断しない



証明書で判断する

Android 9.0のAutofill対策



設計・実装

許可するAutofill Serviceの
証明書ハッシュ値リスト

実行時

現在のAutofill Serviceの
証明書ハッシュ値を確認

アプリ署名検証API

PackageManagerクラス

hasSigningCertificate

Android 9.0 (API Level 28) 以降

アプリ署名検証API

Android 8.1以前

```
public static boolean test(Context ctx, String pkgname, String correctHash) {
    if (correctHash == null) return false;
    correctHash = correctHash.replaceAll(" ", "");
    return correctHash.equals(hash(ctx, pkgname));
}

public static String hash(Context ctx, String pkgname) {
    if (pkgname == null) return null;
    try {
        PackageManager pm = ctx.getPackageManager();
        PackageInfo pkginfo = pm.getPackageInfo(pkgname, PackageManager.GET_SIGNATURES);
        if (pkginfo.signatures.length != 1) return null;    // 複数署名は扱わない
        Signature sig = pkginfo.signatures[0];
        byte[] cert = sig.toByteArray();
        byte[] sha256 = computeSha256(cert);
        return byte2hex(sha256);
    } catch (NameNotFoundException e) {
        return null;
    }
}

private static byte[] computeSha256(byte[] data) {
    try {
        return MessageDigest.getInstance("SHA-256").digest(data);
    } catch (NoSuchAlgorithmException e) {
        return null;
    }
}

private static String byte2hex(byte[] data) {
    if (data == null) return null;
    final StringBuilder hexadecimal = new StringBuilder();
    for (final byte b : data) {
        hexadecimal.append(String.format("%02X", b));
    }
    return hexadecimal.toString();
}
```

Android 9.0以降

```
return pm.hasSigningCertificate(pkgname, Utils.hex2Bytes(correctHash), CERT_INPUT_SHA256);
```

サーバー証明書検証の変更

ホスト名検証に
subjectAltName (SAN)使用

サーバー証明書に
subjectAltNameが必要

SAN必須ブラウザ

Chrome 58　Firefox 48

in the middle attacks. In special cases, it may be appropriate for the client to simply ignore the server's identity, but it must be understood that this leaves the connection open to active attack.

If a subjectAltName extension of type dNSName is present, that MUST be used as the identity. Otherwise, the (most specific) Common Name field in the Subject field of the certificate MUST be used. Although the use of the Common Name is existing practice, it is deprecated and Certification Authorities are encouraged to use the dNSName instead.

https://tools.ietf.org/html/rfc2818

非暗号化通信（HTTP）の抑制

Network Security Configuration

cleartextTrafficPermitted

既定でfalse・HTTP接続不可

やむなくHTTP接続するドメイン**だけ個別**に許可

非暗号化通信（HTTP）の抑制

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <!-- 非暗号化通信はデフォルトで許可しない -->
    <base-config cleartextTrafficPermitted="false">
    </base-config>
    <!-- 例外的に非暗号化通信を許可する必要のあるサイトを、
         <domain-config> タグにより明示的に "true" に設定する -->
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">www.jssec.org</domain>
    </domain-config>
</network-security-config>
```

無指定でも既定でfalse

指紋認証機能

BiometricPrompt API

Android 9.0 (API Level 28)

標準的な認証ダイアログ

今後AndroidXでも対応

指紋認証機能

FingerprintManagerはdeprecated

FingerprintManager

added in API level 23

Deprecated since API level 28

```
public class FingerprintManager
extends Object
```

java.lang.Object
↳ android.hardware.fingerprint.FingerprintManager

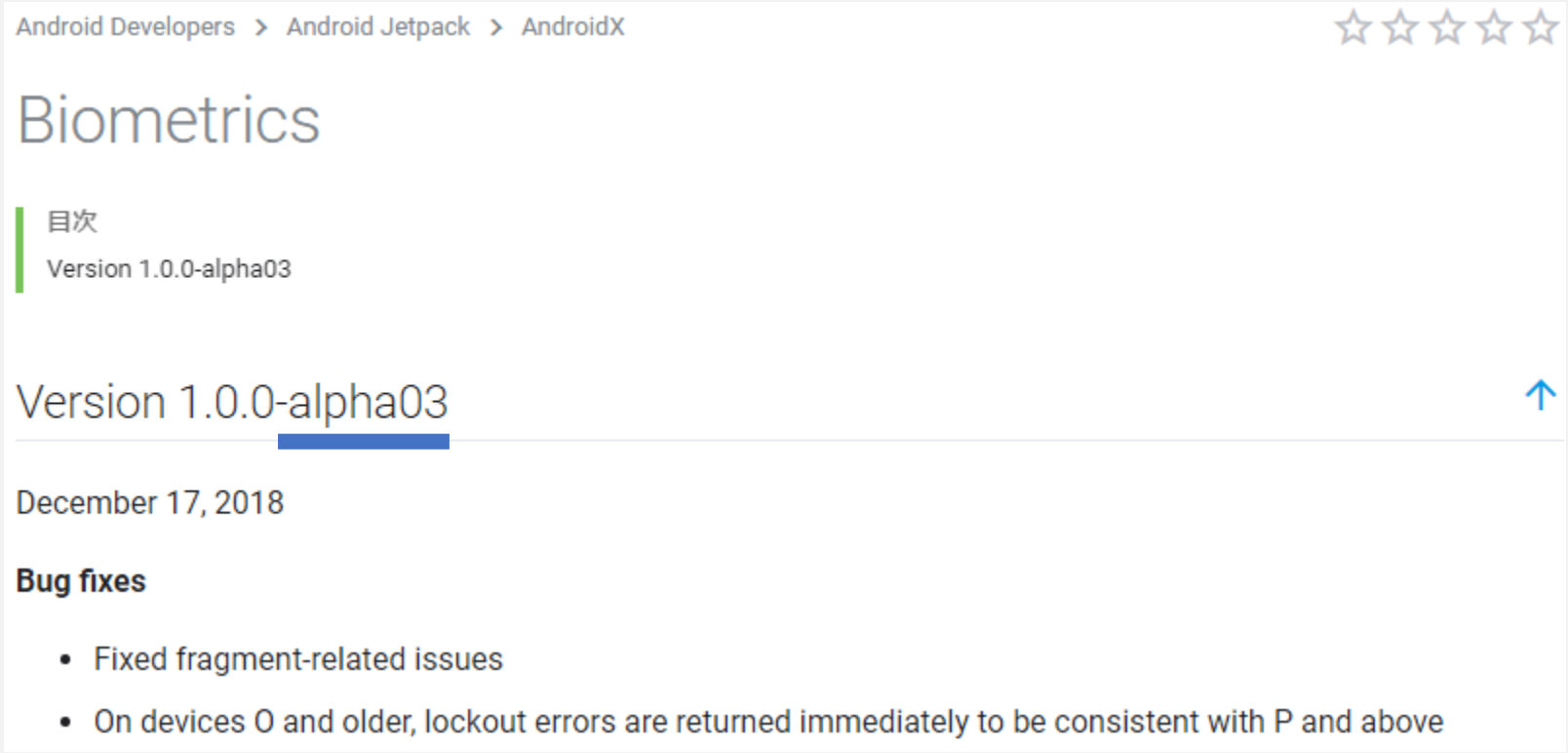
This class was deprecated in API level 28.

See **BiometricPrompt** which shows a system-provided dialog upon starting authentication. In a world where devices may have different types of biometric authentication, it's much more realistic to have a system-provided authentication dialog since the method may vary by vendor/device.

https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager

指紋認証機能

Android 8.1以前はAndroidXで対応



https://developer.android.com/jetpack/androidx/releases/biometrics

指紋認証機能

1. USE_BIOMETRIC パーミッションを利用宣言する
2. "AndroidKeyStore" Provider からインスタンス取得
3. 脆弱でない暗号アルゴリズムで鍵生成
4. 認証の有効期限は設定しない
5. 指紋の登録状況が変わることを前提に設計を行う
6. 暗号化データを指紋認証のみに依存しない

https://developer.android.com/about/versions/oreo

ガイド変更点

Autofillフレームワーク対策
アプリ署名検証API
サーバー証明書検証
暗号化通信
指紋認証機能

Safe Browsing
共有メモリ

Safe Browsing

危険なWebサイトを警告

マルウェア

フィッシング

Safe Browsing

Android 5.0 (API Level 21) ～ 7.0 (API Level 25)

設定　　静的 AndroidManifest.xml

対象　　アプリ全体

Android 8.0 (API Level 26)

設定　　動的 setSafeBrowsingEnabled

対象　　WebViewインスタンス単位

Safe Browsing

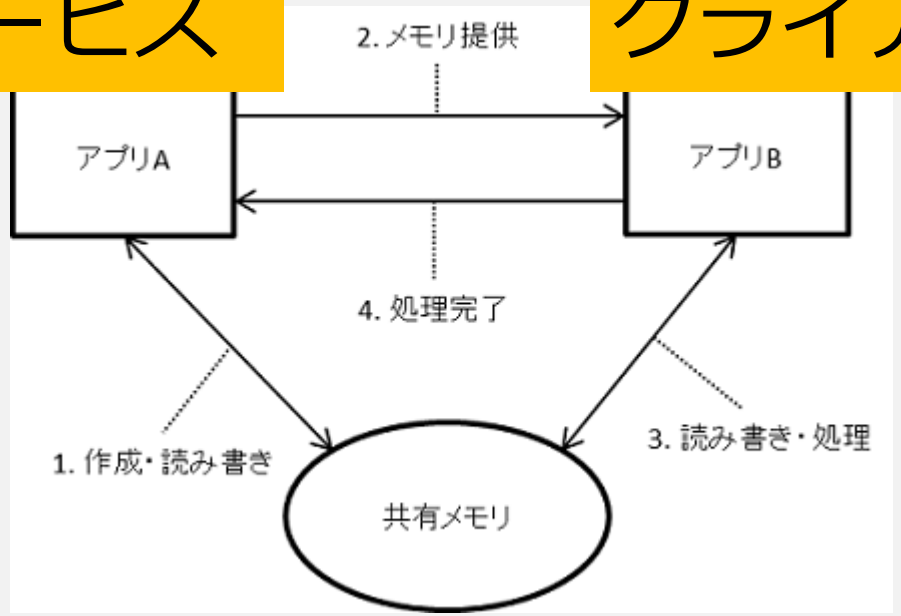
Android OS ver.	Android 標準 WebView	OSとの関係	Safe Browsing
Android 7.0 以降	Chrome for Android (Chromiumベース)	独立	対応
Android 5.0 - 6.0	Android System WebView (Chromiumベース)	独立	対応
Android 4.4	OS組み込み WebView (Chromiumベース)	一体	非対応
Android 4.3 以前	OS組み込み WebView	一体	非対応

共有メモリ

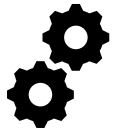
アプリの間で同一の物理メモリ領域を共有する機構

共有メモリ提供サービス

クライアント



共有メモリ



共有メモリ提供Serviceを実装

「Serviceを作る・利用する」に準ずる

共有メモリ

分類	非公開Service	公開Service	パートナー限定Service	自社限定Service
startService型	-	-	-	-
IntentService型	-	-	-	-
local bind型	o	-	-	-
Messenger bind型	o	o	-	o*
AIDL bind型	o	o	o	o

共有メモリ

S1. Service は SharedMemory.create() により共有メモリを作成する

S2. Service 自身が共有メモリを利用する場合 SharedMemory#map() によって自身のメモリ空間に共有メモリをマップする

C1. クラアントは明示的 Intent を使用し Context#bindService() によってサービスに接続する

S3. クライアントからの接続要求が届くと、Service のonBind() call back が呼び出される。Service はここで(あれば)必要な前処理を行い、クライアントへ接続用の IBinder を返す。

C2. Service が onBind() を実行した際の戻り値(IBinderのインスタンス) は、クライアント側の onServiceConnected() callback の引数として返される。以降はこの IBinder を利用して Service との通信を行う。

C3. クライアントは Service に対して共有メモリを要求する。

S4. Service はクライアントからの共有メモリ要求を受け、クラアントが共有メモリをアクセスする際に許可する操作(読み・書き)を設定する。

S5. Service はクライアントへ共有メモリオブジェクトを渡す。

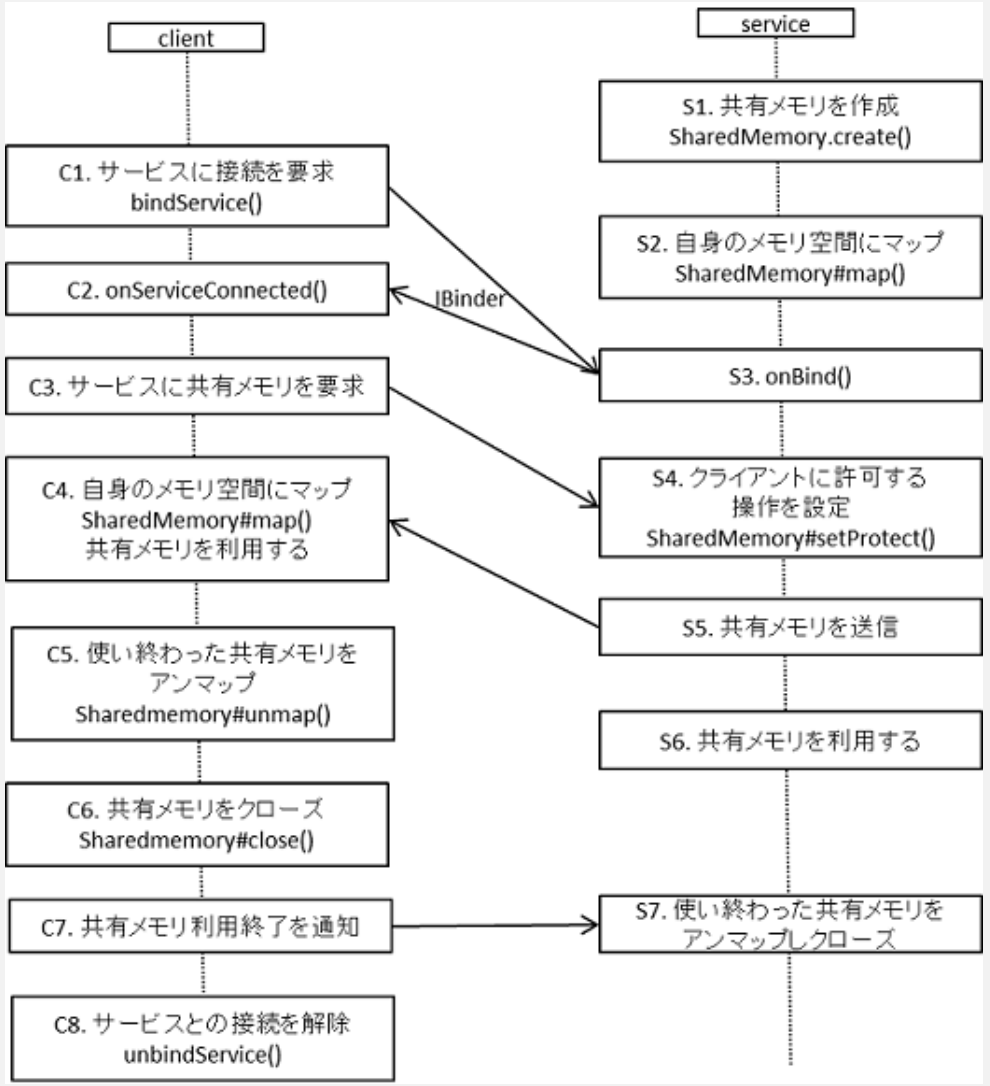
C4. クライアントは受け取った共有メモリにアクセスするため、自身のアドレス空間へ共有メモリをマップし、利用する。

C5, C6. クライアントが共有メモリ利用が終了したとき、自身のメモリ空間からアンマップし(C5)、共有メモリをクローズ(C6)する。

C7. クライアントはその後、共有メモリの利用が終了したことをサーバーへ通知する。

C8. クライアントは Service との接続を解除する。

S7. Service はクライアントから利用終了のメッセージを受け取ったのちに、自身も共有メモリをアンマップしクローズする。



TLS1.2への移行

Android OS ver.	Android 標準 WebView	OSとの関係	TLS1.2
Android 7.0 以降	Chrome for Android (Chromiumベース)	独立	対応
Android 5.0 - 6.0	Android System WebView (Chromiumベース)	独立	対応
Android 4.4	OS組み込み WebView (Chromiumベース)	一体	対応（既定で無効）
Android 4.3 以前	OS組み込み WebView	一体	非対応

WebView TLS1.2対応　≧Android 4.4

Android OSシェア　≧Android 4.4　97%

≧Android 5.0　89%

2018年10月26日時点

https://developer.android.com/about/dashboards/index.html

次版ガイド作成

次版ガイド作成

次のAndroid OSに合わせて発行予定
ご協力してくださるかた募集中
参加方法はWG（メーリングリスト）参加

セキュアコーディングWG募集中

1. JSSECまたは Androidセキュリティ部に参加

JSSEC会員か確認

http://www.jssec.org/members/

Androidセキュリティ部に参加

https://groups.google.com/group/android-security-japan

ご協力お願い致します

2. 参加メール送信

To: Jun.Ogiso@sony.com
Subject: JSSECセキュアコーディングG参加
――――――――――――――――――――――――――
(1) Google account:
(メールアドレス)
(2) First name:
(名)
(3) Last name:
(姓)
(4) Organization:
JSSEC参加組織名 or Androidセキュリティ部
(5) Git access:
(必要 or 不要)

各種アカウント発行後メールでご連絡します