セキュアコーディングガイド Android 6.0対応

2016年2月24日

JSSEC技術部会
副部会長　奥山　謙
（ソニーデジタルネットワークアプリケーションズ株式会社）
Ken.Okuyama@jp.sony.com

アジェンダ

- セキュアコーディングガイドの歴史
- 〃　第5版の改定内容
- 〃　第6版の改定内容

セキュアコーディングガイドの歴史

Androidアプリセキュリティの教科書

Androidアプリセキュリティのノウハウ集

通称：JSSECセキュアコーディングガイド
PDF文書とセキュアなサンプルコード一式（無償）
http://www.jssec.org/report/securecoding.html
「Android セキュアコーディング」と検索

デファクトスタンダードなガイド・基準

総務省も推奨のガイド。
通信キャリアや
多くのアプリベンダーでも活用。
受入基準にするアプリ発注会社もある。

総務省 MIC Ministry of Internal Affairs and Communications

http://www.soumu.go.jp/menu_news/s-news/01ryutsu03_02000043.html

ガイドの歴史

- 年1回から2回のペースで改訂

2012年6月

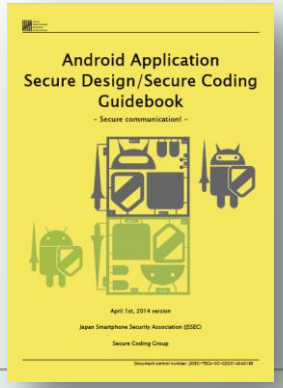
2012年11月

2013年4月

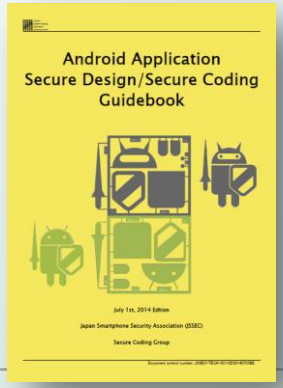
2014年7月

- 2014年、英語版リリース

2014年4月



2014年8月



2015年6月1日版 (第5版)

- 2015/7/10にセキュアコーディングガイド 2015年6月1日版 (第5版)を一般公開
- 改定内容
  - 最新の統合開発環境 Android Studio に対応
  - Android Lollipop に対応

※EclipseプラグインベースのADT (Android Development Tools)は 2015年12月にサポート打ち切り

2016年2月1日版 (第6版)

- 2016/2/3にセキュアコーディングガイド 2016年2月1日版 (第6版)を一般公開
- 改定内容
  - Android 6.0対応
    - Permissionモデルの変更に対応
    - 指紋認証
  - Android 5.0以前の項目追加
    - 5.0で刷新された Notification
    - getRunningAppProcess()仕様変更
    - WebViewに追加された機能の注意点 など

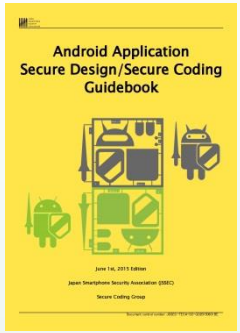
セキュアコーディングガイドの第5版の改定内容

第5版の特徴

２０１５年６月１日版　改定内容

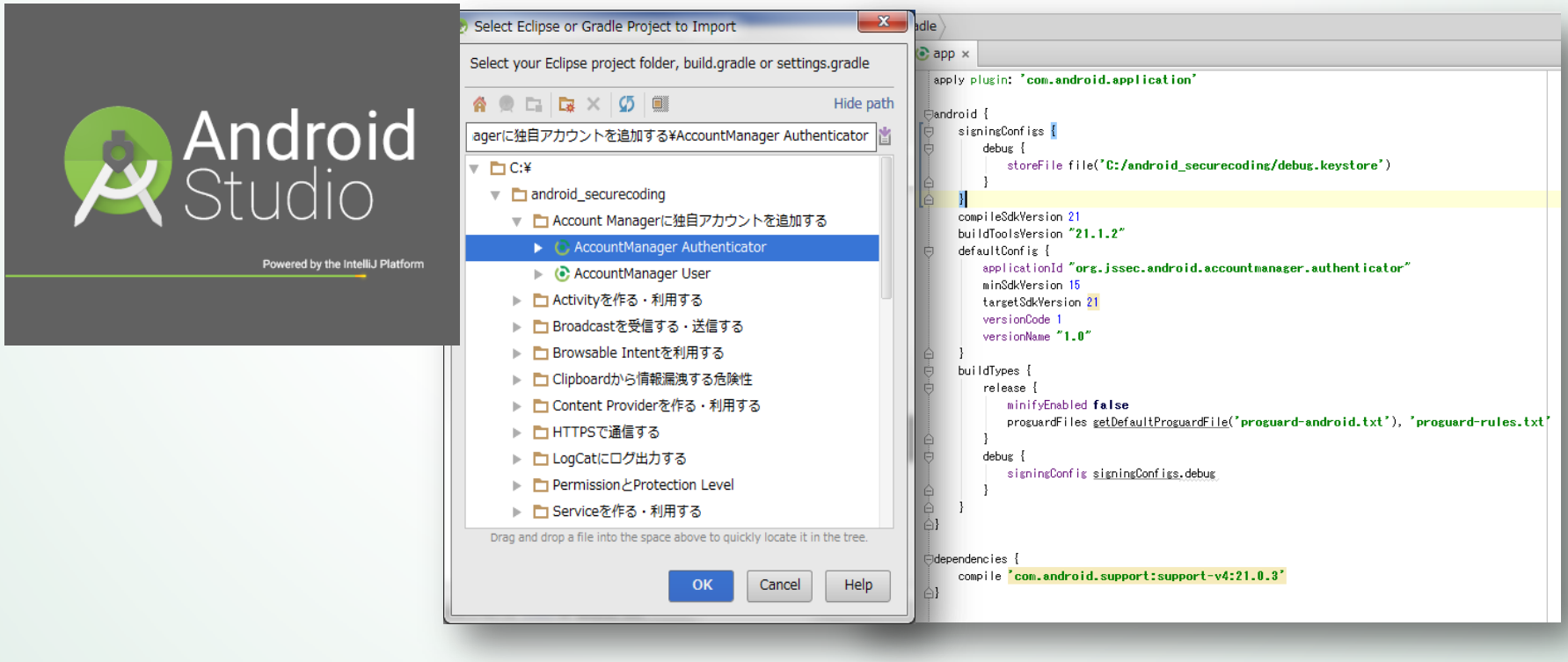
下記の内容で記事を更新しました

- 最新の統合開発環境 Android Studioに対応
- 既存記事をAndroid Lollipopまでの変更に対応



日本語版・英語版　同時公開

Android Studio対応



- ✓ セキュアなサンプルコードをAndroid Studio 上で活用する方法を解説
- ✓ 署名付きアプリの生成方法など、Android Studioの使い方も解説

既存記事のLollipop対応

- 既存記事を最新のLollipop環境に対応
  - 新しいセキュリティ問題やコーディング方法を追記

例えば・・・

WebView

- CVE-2013-4710
  - 執筆当時ゼロデイだったため、記載できなかったので今頃対応
- FileスキームのHTMLから任意のローカルファイルにアクセスできる件

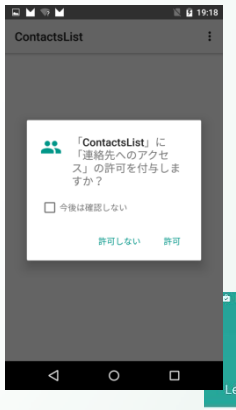
HTTP通信

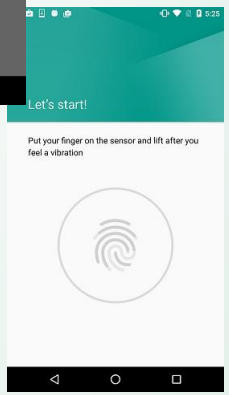
- 本版で対応API level 15以降を対象と改めたことを受けApache HttpClientライブラリを利用した方法から、HttpURLConnectionを利用する方法に変更
  （以前は API Level 8以上を対象としていました）

セキュアコーディングガイドの第6版の改定内容

第6版 の特徴

- Android 6.0対応
  - Permissionモデルの変更に対応
    - permission の protectionLevel 変更
      - ネットワーク系のpermissionは軒並みnormalへ
    - 新しいPermission Model
      - アプリ実行中に必要なタイミングでユーザー許諾を得る
  - 指紋認証を使ったアプリ





- Android Lまでの新規項目追加
  - Lollipopで刷新された Notification まわり
  - getRunningAppProcess()仕様変更
  - WebViewに追加された最新機能の注意点

以下の項目をご紹介します

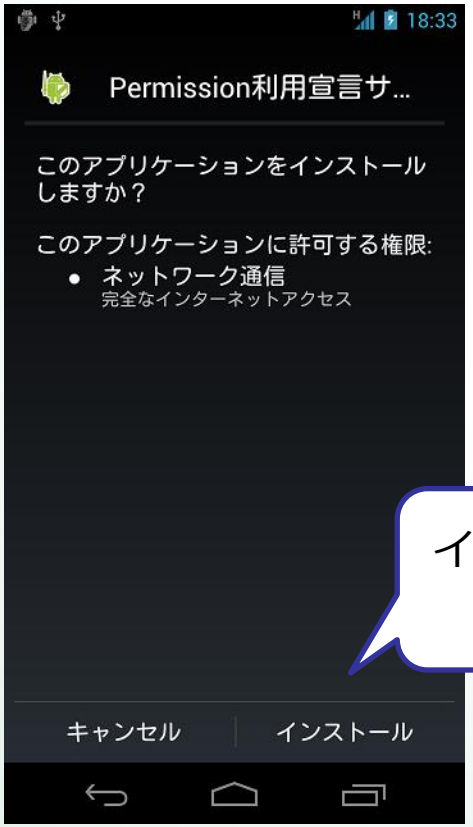
1. 新しいPermissionモデル
   (Runtime Permission)

2. 指紋認証によるユーザー認証

3. Notification Visibility

新しい Permissionモデル (Runtime Permission)

新しい Permission Model



インストール時に確認

実行時に確認

従来のPermission確認画面

新しいPermission確認画面

新しい Permission Model

- 保護レベルがdangerousのpermissionのみ影響
  - ① 実行中の**必要に応じたユーザー許諾**
  - ② **Permission Group単位での権限管理**
    同じPermission Groupに属するpermissionは一度の要求で全てgrantされる
  - ③ **ユーザーによる許可の取り消し (revoke)**
    - targetSdkがAndroid Mより前のバージョンでも、
      Android M 端末上では、ユーザーは権限をrevokeできる
    - Android Mより前のOSの動作する端末上では新しいモデルは適用されない

権限要求用のAPIを呼び出す

- 以下のメソッドで権限を要求する
  - Activity.requestPermissions(String[], int)
  - Fragment.requestPermissions(String[], int)
    - 第一引数で必要なpermissionを列挙
    - 第二引数でコールバックに渡すリクエストコードを指定
      - startActivityForResult →onActivityResult とよく似ています

```
// READ_CONTACTSを要求する例
requestPermissions(
    new String[] {Manifest.permission.READ_CONTACTS},
    REQUEST_CODE);
```

こんな感じのダイアログが出ます。
レイアウトは変更できません

結果をコールバックで受け取る

- 権限要求用APIを呼び出した Activity/Fragment のコールバックで結果を受け取れます
  - Activity.onRequestPermissionsResult
  - Fragment.onRequestPermissionsResult
    - 引数にそれぞれ、リクエストコード、要求した権限(String配列)、結果(Granted/Denied)が渡される

ユーザーの選択後に呼び出される。
「許可」した場合、
permissions = [READ_CONTACTS],
results = [PERMISSION_GRANTED]
となる。

```
// READ_CONTACTSの例
public void
    onRequestPermissionsResult(int reqCode,
        String[] permissions, int[] results) {
  // 結果をみて、処理を進める
}
```

(最低限度) Android 6.0に対応するために

- Android Mの端末上で動作する (落ちない) アプリにするために、最低限下記の対応が必要となります
- dangerousレベルのpermissionが必要なアプリは
  - Activity, Fragmentで
    - requestPermissionsを呼び出し
    - onRequestPermissionsResultをオーバーライドする
  - Service, Providerなどで 権限の必要なAPIを呼び出す場合
    - Context.checkSelfPermissionで権限が付与されているか確認してからAPIを呼ぶ
      - Providerの場合はgetContext().checkSelfPermission(…)

指紋認証による
ユーザー認証

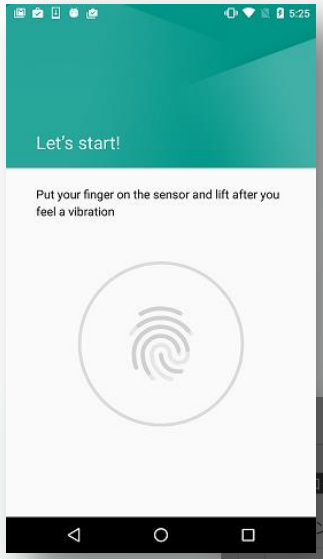
指紋認証(Fingerprint)

- Android 6.0 (API Level 23)以降
  - 22以前との互換性は、FingerprintManagerCompatクラスを
    使用することで吸収可能
- Fingerprint機能が端末にないと使用不可
  - FingerprintManager.isHardwareDetected()
- ロックが設定されていないと使用不可
  - KeyguardManager.isKeyguardSecure()
  - Fingerprintが登録(Enroll)できない
- 指紋が登録されていないと使用不可
  - FingerprintManager.hasEnrolledFingerprints ()

アプリが指紋認証を使う目的

アプリが端末ユーザーを「簡易的に」認証するため

- 端末ユーザーのみ実行可能な機能を保護したい
  - 機能実行時の認証
- 端末ユーザーのみ実行可能な機能の操作を簡略化したい
  - 機能実行時の認証の簡略化
  - 例：楽天銀行アプリがログインに使用



指紋認証の特徴

ID/Passwordに比べて

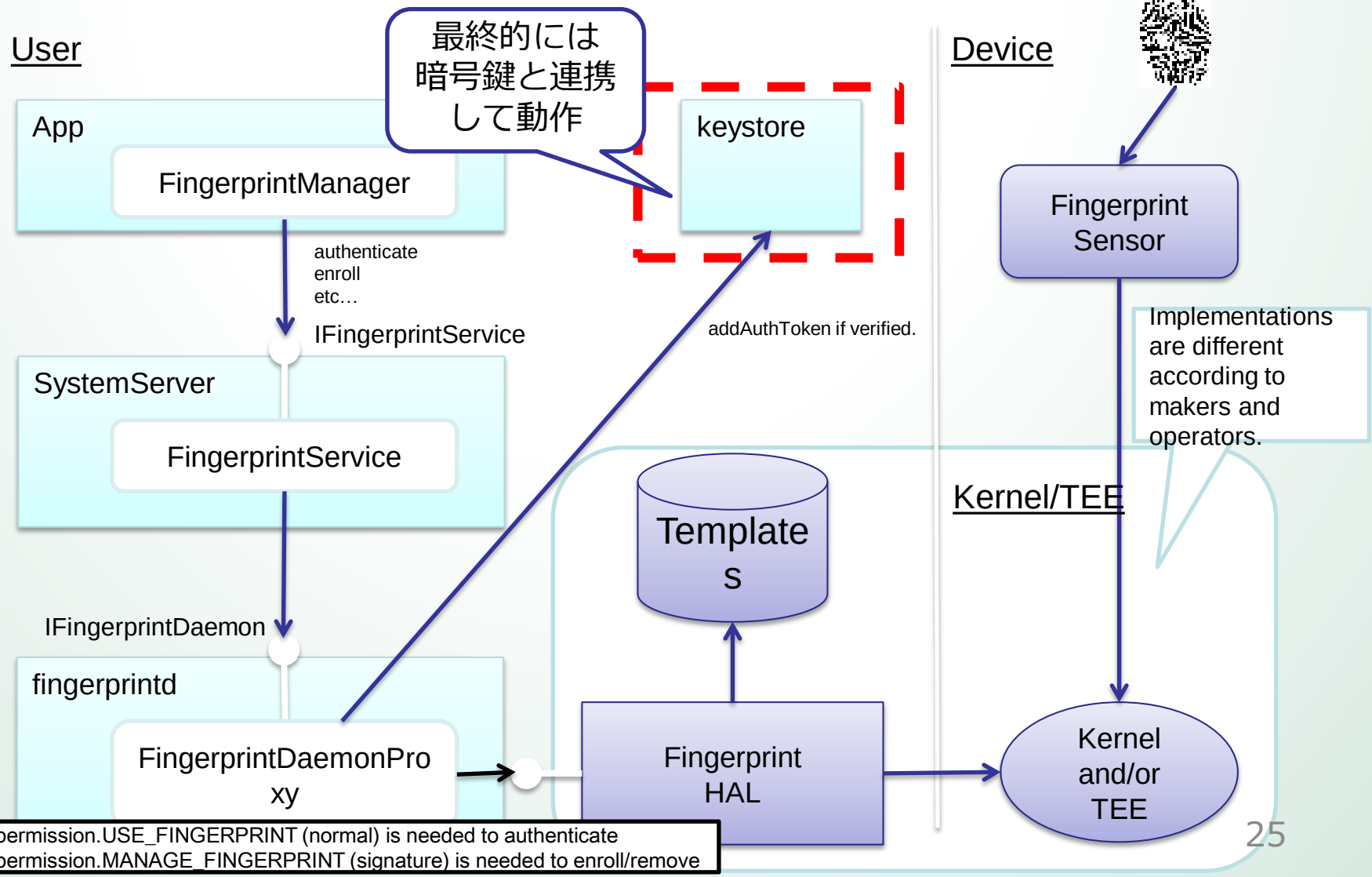
[良い点]

- 入力が容易(覚える必要もない)
- 端末へのアクセスが必要（攻撃を受けにくい）

[悪い点]

- 認証としての強度は（実は）弱い
  - 秘匿困難性、人工物による認証、認証精度
- 指紋が漏れてても変えられない
  - 非交換性
- 生体変化や環境条件による識別精度の低下
  - パスワードのように一意に決まらない
  - ケガなどで認証ができなくなる可能性

Fingerprint Architecture



android.permission.USE_FINGERPRINT (normal) is needed to authenticate
android.permission.MANAGE_FINGERPRINT (signature) is needed to enroll/remove

（参考）AndroidKeyStore

- 機能
  - 鍵の管理(KeyStore)：
    Import/Export (Store/Load)
  - 鍵の生成(KeyGenerator)
  - 鍵の使用(Cipher/Signature/Mac)：
    暗号化・復号、署名・検証

- 指紋認証との関係
  - 上記で暗号鍵を生成 or 既存の鍵を登録するとき
  - 生成 or 登録以後に鍵を使うに際の条件として
  - 指紋認証（で認証すること）を条件に設定できる

アプリが指紋認証を使うときは…

- 指紋認証の弱い点を考慮する
  - 指紋認証だけに頼る設計にはしない
    - 認証ができなくなった（生体が変化するなど）でも、別手段で機能を利用できるようにする
  - 機密性の高い情報・機能は（直接）扱わない
    - 課金や決済といった重要な情報・機能を扱う際はかならず他の認証方法と併用する
- 正しい暗号の使い方も理解する
  - 指紋認証は暗号処理と密接に関連するため、正しい暗号の使い方も理解しておく必要がある
    - アルゴリズム選択、鍵長の設定、端末内での鍵管理方法、暗号機能自体の認証・・・

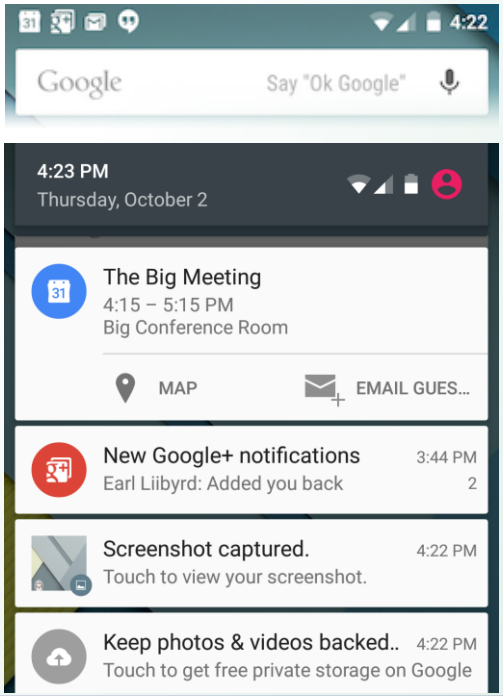
Notification Visibility

今どきのNotification（通知）

画面ロック中にもNotification（通知）表示

Notificationが表示

画面ロック中



メール着信などが分かり利便性が上がりました

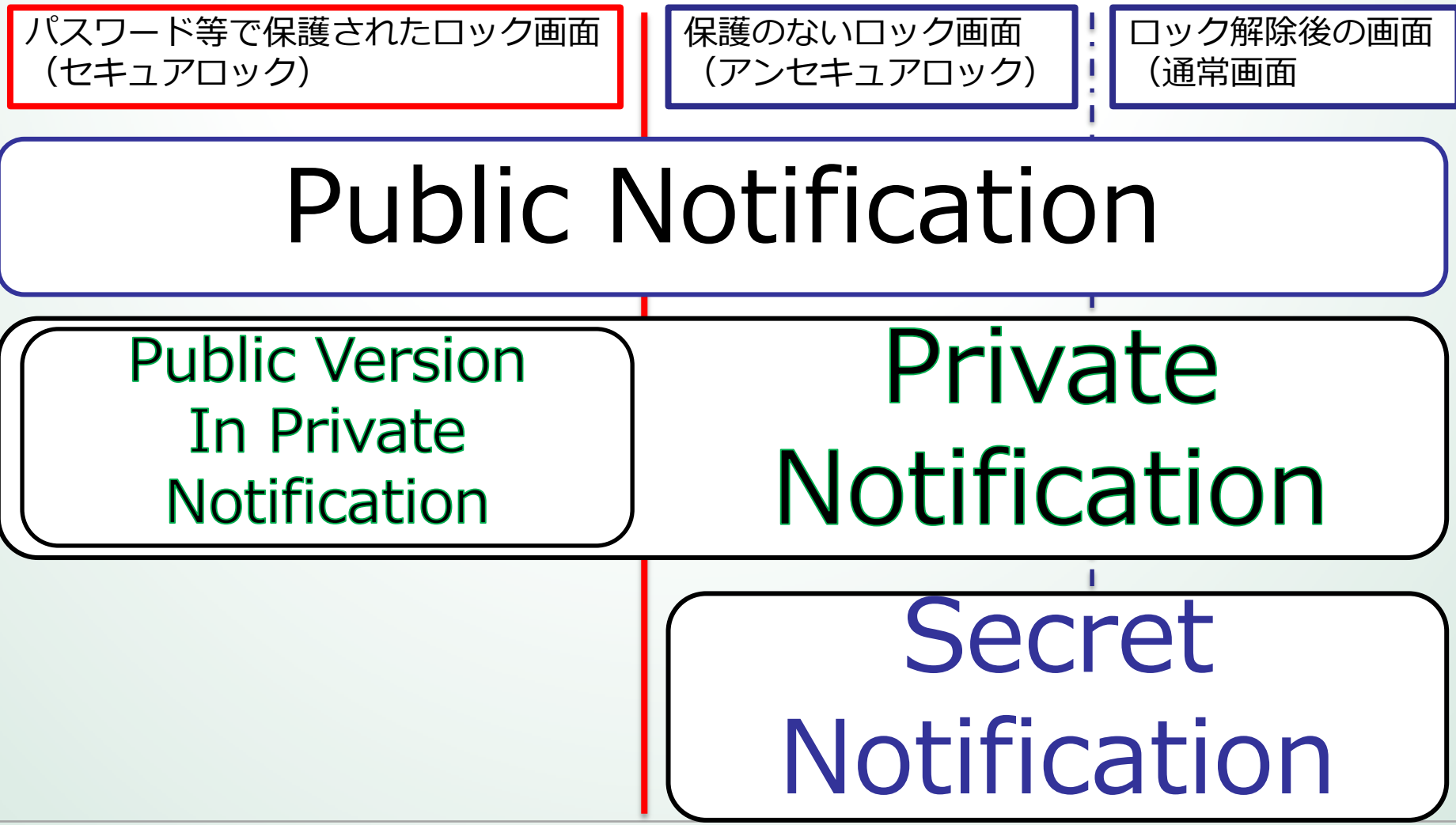
Notification Visibility

Notificationを表示する条件設定

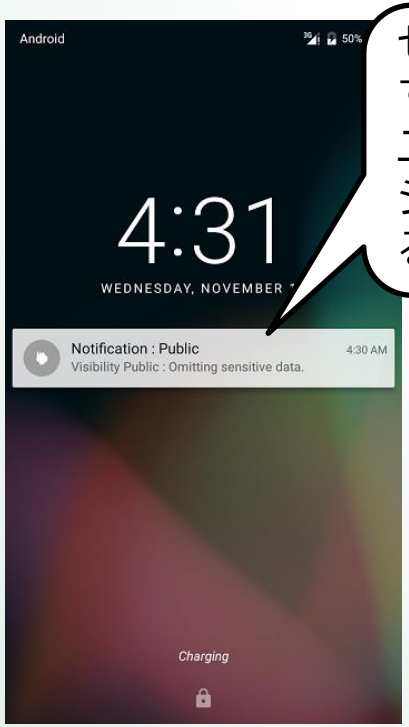
Visibilityの値	Notificationの振る舞い
Public	すべてのロック画面上でNotificationが表示される
Private	すべてのロック画面上でNotifcationが表示されるが、パスワード等で保護されたロック画面（セキュアロック）上では、Notificationのタイトルやテキスト等が隠される（プライベート情報が隠された公開可能な文に置き換わる）
Secret	パスワード等で保護されたロック画面（セキュアロック）上では、Notificationが表示されなくなる（セキュアロック以外のロック画面ではNotificationは表示される）

※ Android 6.0 現在のVisibilityのデフォルト値は「Private」

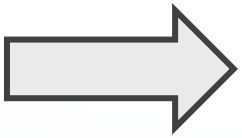
Notification Visibility



Notificationのセキュリティ



セキュアロックで表示するNotificationに、ユーザーのプライバシー情報が含まれていると・・・



個人情報漏えい

Notificationが画面ロック中にも表示されるようになったことで新たなセキュリティ問題が発生する可能性が生まれた

Notificationの注意点

- Visibilityと通知内容
  - 表示しようとしている情報は設定したVisibilityに沿った内容であるか確認
  - デフォルト値に頼らず明示的にVisibilityを設定
- Private Notificationの使い方
  - Private Notificationは内部にPublic Versionを保持
  - 画面ロック中にユーザーに見せる情報と、ロック解除後に見せる情報を正しく切り分け・設定すること
    - 例：メール情報
      画面ロック中：メール通知があったことのみを通知
      ロック解除後：メールの本文や送信者などの情報を通知（可能）

さいごに

執筆へのご協力をお願いします

ご協力いただける方は下記の要領でご連絡ください

1. JSSEC会員でない方はAndroidセキュリティ部へご参加ください

- 会員かどうかは下記URLで確認
  - http://www.jssec.org/members/
- Androidセキュリティ部への参加は下記URLからできます
  - https://groups.google.com/group/android-security-japan
  - 右の(4)では「Androidセキュリティ部」と記載してください

2. 次の書式でメール送信してください

```
To: Masaru.Matsunami@jp.sony.com
Subject: JSSECセキュアコーディングG参加
本文:
(1) Google account: (メアドを記載)
(2) First name: (名前を記載)
(3) Last name: (名字を記載)
(4) Organization: (組織名を記載)
(5) Git access: (必要 or 不要)
```

- 各種アカウント発行後、メール返信にてご連絡いたします

  (1)はGmailメアドまたはご自身のメアドをGoogleアカウント化したもの

  (4)は下記URLページ内から選択
  - http://www.jssec.org/members/

ご協力お願い致します