IPA Technical Watch

IPA

IPA テクニカルウォッチ

「Android アプリの脆弱性」に関するレポート

～簡易チェックリストで脆弱性を作り込みやすいポイントを確認しましょう～

IPA 独立行政法人情報処理推進機構
技術本部 セキュリティセンター

目次

目次 1
はじめに 2
本書の対象読者 2
本書の構成 2
本書における用語 2
1. Android アプリの概要 4
1.1. Android アプリの特徴 4
2. Android の仕組み 6
2.1. Android の仕組みの概説 6
2.2. アプリの動作に必要な権限の許可 7
2.3. ファイルのアクセス許可 8
2.4. 他のアプリとのデータ共有 10
2.5. 他のアプリとの機能連携 11
3. IPA に報告された Android アプリの脆弱性 13
3.1. 脆弱性の傾向 13
3.2. 脆弱性が作り込まれた原因の考察 13
4. 脆弱性例の紹介 14
4.1. ファイルのアクセス制限不備の脆弱性 14
4.2. コンポーネントのアクセス制限不備の脆弱性 17
4.3. ログ出力に関する情報漏えい 20
5. 簡易チェックリスト 22
おわりに 23

はじめに

スマートフォンの利用者は年々増加している。2011 年のスマートフォンの出荷台数は 2,300 万台、スマートフォン OS 別契約数シェアは Android OS が過半数を超えたという調査結果がある[1]。

そのような状況の中、**IPA に届け出られる Android アプリの脆弱性関連情報も増加傾向にある。届け出られた脆弱性は、アクセス制限の不備に関するものが 7 割以上であった。さらに分析した結果、これらは Android の仕組みを理解し、適切にアクセス制限の設定をしていれば防ぐことのできる脆弱性であることがわかった。**しかし、多くの届出があったという事実から、この Android 特有の設定内容が開発者に周知できておらず、結果的に、アクセス制限の不備の脆弱性を作り込んでしまっているのではないかと IPA では推測した。

本レポートでは、Android 特有のアクセス制限の必要性を理解いただくために、Android の仕組みについて説明した上で、脆弱性を作り込まないポイントを交え、届出の多かった Android アプリの脆弱性の例を 5 つ紹介している。また、紹介した脆弱性の確認項目をリスト化した簡易チェックリストを掲載している。簡易チェックリストを使用して、自組織で開発したアプリに脆弱性が作り込まれていないか確認いただきたい。

本書が、Android アプリに作り込みやすい脆弱性の把握と対策方針の参考となれば幸いである。

本書の対象読者

本書の対象読者は、Android アプリの企画・開発に携わる方や、これから Android アプリを開発しようとしている方を想定している。

本書の構成

本書は 5 つの章で構成している。Android アプリの脆弱性の傾向と対策を把握されたい方は、P.3 の「本書における用語」を確認した上で、3 章および 4 章を参照いただきたい。

章	章から分かること
1. Android アプリの概要	● Android アプリの特徴
2. Android の仕組み	● Android OS が提供する仕組みの概要
3. IPA に報告された Android アプリの脆弱性	● IPA に届け出られた Android アプリの脆弱性の傾向 ● 脆弱性が作り込まれた原因の考察
4. 脆弱性例の紹介	● Android アプリに作り込みやすい脆弱性と対策のポイント
5. 簡易チェックリスト	● 作り込みやすい脆弱性のチェックポイント

なお、本書では Android アプリの具体的なコーディングについては触れていない。コーディング例は、日本スマートフォンセキュリティ協会（JSSEC）が公開している『Android アプリのセキュア設計・セキュアコーディングガイド』[2]などをご参照いただきたい。

---

[1] 株式会社 MM総研: スマートフォン市場規模の推移・予測（12 年 3 月）
http://www.m2ri.jp/newsreleases/main.php?id=010120120313500 〔2012 年 5 月末日時点〕

[2] 『Android アプリのセキュア設計・セキュアコーディングガイド』【6 月 1 日版】
http://www.jssec.org/dl/android_securecoding.pdf 〔2012 年 6 月 12 日時点〕

本書における用語

■ 「Android OS」

米国 Google（グーグル）社が提供する OS（基本ソフト）。オープンソースソフトウェアの「Linux」などを基に開発され、スマートフォンやタブレットなどに採用されている。

■ 「Android アプリ」

Android OS 上で動作するアプリケーションを指す。Android アプリは、Android OS の仕組みの上で動作する。

■ 「脆弱性」

Android OS に備わっている仕組みを"Android アプリが適切に使用していない"場合、他のアプリが所有しているデータに不正にアクセスしたり、他のアプリの権限を不正に使用したりするセキュリティ上の問題が発生する。本書では、このようなセキュリティ上の問題を Android アプリにおける脆弱性と位置づけている。

■ 「不正なアプリ」

本書では、他の Android アプリの脆弱性を悪用し、他アプリのデータにアクセスしたり、他アプリの権限を不正に使用したりする Android アプリを指す。

■ 「コンポーネント」

Android アプリを構成する要素のこと。下記 4 種類が存在する[3]。

コンポーネント	説明
アクティビティ	電話をかけたり、写真を撮ったり、メールを送るなどのユーザインターフェースを提供するコンポーネント。Android アプリは、基本的に複数のアクティビティで構成されている。
サービス	バックグラウンドで処理を継続するコンポーネント
コンテントプロバイダ	Android アプリのデータを他のアプリに公開するコンポーネント
ブロードキャストレシーバ	Android OS や他のアプリが発信する通知を受信するコンポーネント

[3] Application Fundamentals | Android Developers
http://developer.android.com/guide/topics/fundamentals.html#Components 〔2012 年 5 月末日時点〕

1. Android アプリの概要

本章では、アプリ間でのデータの共有や連携、カメラや GPS などの端末に備わった機能の使用といった、Android アプリの特徴を説明する。

Android OS は、PC のようにアプリケーション（Android アプリ）をインストールすることで、様々な用途での端末の利用を可能としている。Android アプリがどのようなデータや機能を使用するのか見てみよう。

1.1. Android アプリの特徴

■ 各種データにアクセスできる

Android アプリは、端末に保存されたデータにアクセスすることができる（図 1-1）。基本的に、データはアプリ毎に用意された専用の領域（/data/data/<アプリのパッケージ名>）、または SD カードに保存されている。

なお、通常は他のアプリの専用領域にアクセスすることはできないが、Android の仕組みを使用することにより、アクセスできるようになる[4]。

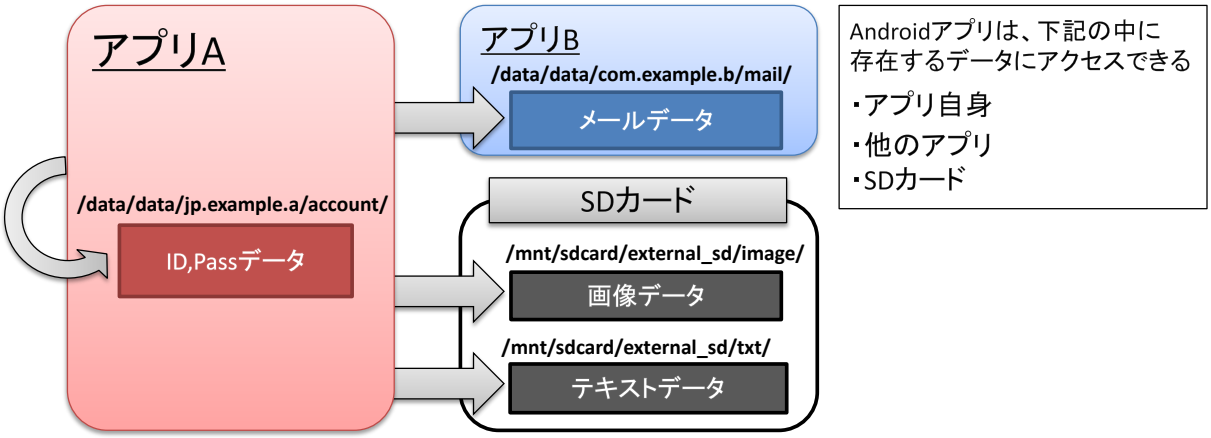


図 1-1 Android アプリのデータアクセスイメージ

■ 各種機能にアクセスできる

Android アプリは、OS や他の Android アプリの機能を使用することができる（図 1-2）[5]。これにより、Android アプリからカメラで写真を撮影したり、位置情報を取得したり、他のアプリとの連携が可能となる。例えば、アプリ A がカメラ機能を使って撮影した写真を加工し、その画像データをアプリ B でメール送信するといったことが可能である。

[4] Android の仕組みについては 2 章で説明する。

[5] 各種機能にアクセスするためには、機能を使用する側のアプリは必要な権限を取得する必要があり、機能を提供する側のアプリは他のアプリにアクセスを許可する必要がある。この権限取得や機能のアクセス許可の設定も Android アプリの仕組みを使用する。

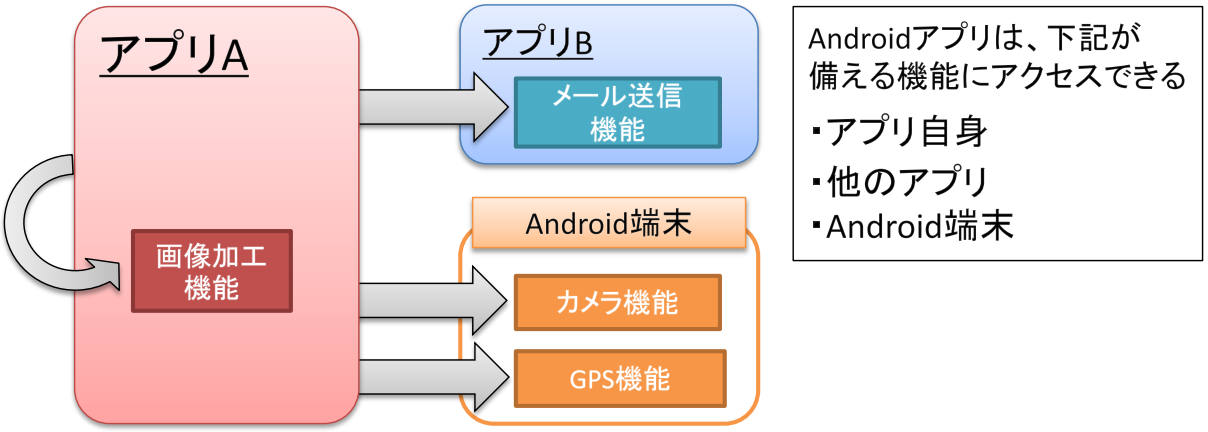


図 1-2 Android アプリの機能アクセスイメージ

■ プライバシーに関する情報が蓄積されやすい

Android 端末などのスマートフォンは高機能かつ携帯性に優れた端末である。そのため、端末を常用していると、電話の通話履歴やインターネットの閲覧履歴、アドレス帳データ、位置情報、カメラ機能で撮影された動画や静止画のデータなどといった、いわゆるプライバシーに関する情報（図 1-3）が蓄積されやすい。

図 1-3Android 端末が取り扱う情報例

従って、Android アプリは Android の仕組みを使用して、アプリが保有するデータや機能を適切に管理する必要がある。

2. Android の仕組み

本章では、Android OS が提供する仕組みの一部について解説する。Android アプリはこの仕組みの上で動作している。Android OS がどのような仕組みを提供しているのか見てみよう。

2.1. Android の仕組みの概説

Android OS は、「Android アプリに必要以上の権限を持たせないように制限するための仕組み」と「他のアプリとデータ共有や機能連携するための仕組み」を備えている（図 2-1）。

前者の仕組みにより、Android アプリは特に権限を要求しない限り、基本的に他のアプリのデータや機能などへのアクセスができない。アクセスするためには、必要な権限を取得し、さらに後者の仕組みを活用する必要がある。その結果、1 章「Android アプリの概要」で説明したように、Android アプリは様々な機能やデータへアクセスできるようになる。

これら Android の仕組みを適切に使用することで、アプリが保有するデータや機能を適切に管理することができる。次節の 2.2、2.3 において前者に仕組みを、2.4、2.5 において後者の仕組みを説明する（表 2-1）。

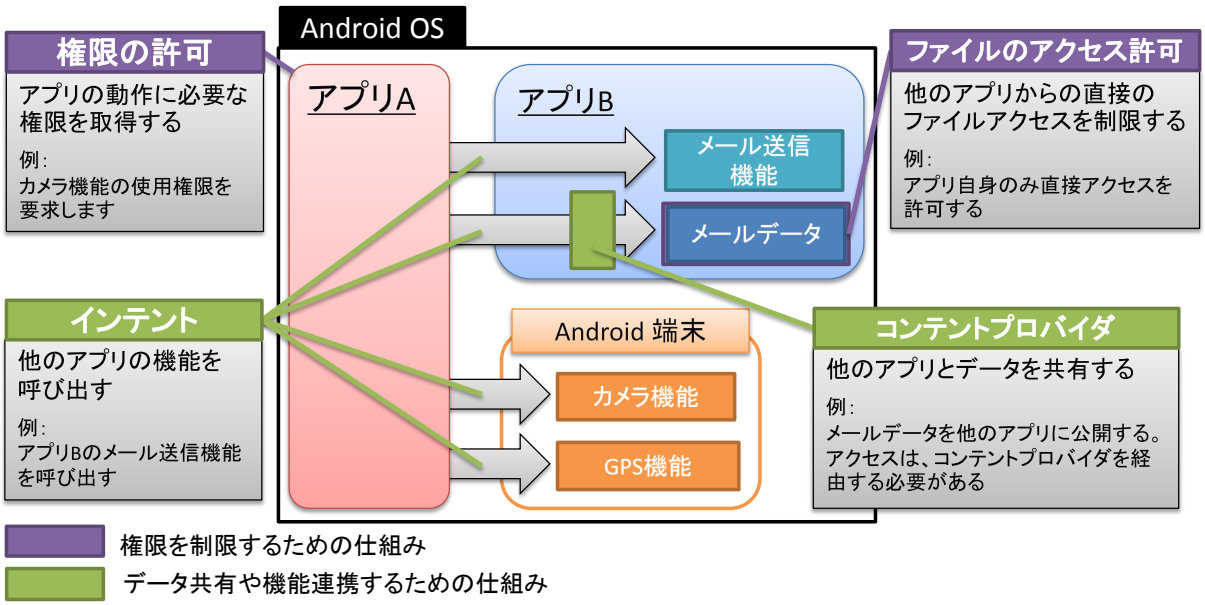


図 2-1 Android の仕組みの動作イメージ

表 2-1 Android の仕組み一覧（一部）

種類	名称	用途
権限を制限するための仕組み	権限の許可	アプリの動作に必要な権限を取得する（2.2 節参照）
	ファイルのアクセス許可	他のアプリからの直接のファイルアクセスを制限する（2.3 節参照）
データ共有や機能連携するための仕組み	コンテントプロバイダ	他のアプリとデータを共有する（2.2 節参照）
	インテント	他のアプリの機能を呼び出す（2.5 節参照）

2.2. アプリの動作に必要な権限の許可

Android OS の仕組み上、Android アプリは、アプリの動作に必要な権限をアプリインストール時に取得しなければならない。例えば、Android アプリから電話発信したい場合や位置情報を取得したい場合、他のアプリの機能を使用したい場合には、それに応じた権限が必要である。使用したい機能やデータに対する権限がなければ、当該箇所の処理は正常に実行されない。なお、独自の権限を定義することもできる。

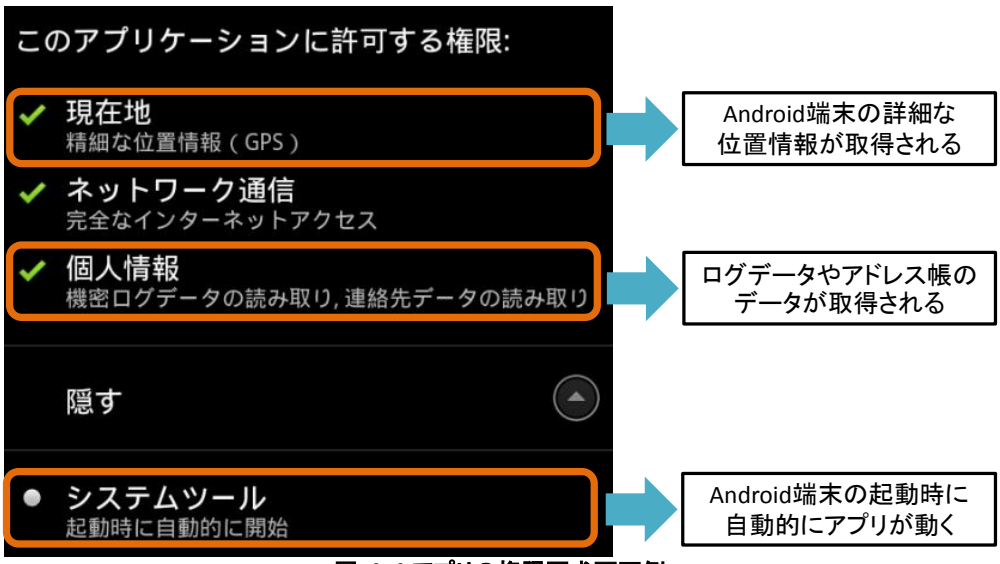


図 2-2 アプリの権限要求画面例

前述した通り、アプリが権限を取得するタイミングは当該アプリのインストール時である。アプリのインストール時に、Android OS により、当該アプリの AndroidManifest.xml[6]ファイルに記述した権限を許可するか否かを確認する画面が表示される（図 2-2）。利用者はアプリが要求している権限を確認し、最終的に許可（＝インストール）するかどうかを決める。

なお、利用者を騙して直接情報窃取するタイプのアプリは様々な権限を要求する。そのため、Android アプリ開発者は、悪意あるアプリと疑われないためにも不要な権限を要求しないように注意する必要がある。下記に権限の例を紹介する。

権限名	説明
INTERNET	アプリがインターネットに接続することを許可する
READ_CONTACTS	アドレス帳に登録した連絡先の読み取りを許可する
READ_LOGS	システムログファイルの読み取りを許可する
ACCESS_FINE_LOCATION	GPS などを使用した、端末の詳細な位置情報へのアクセスを許可する
ACCESS_COARSE_LOCATION	携帯電話の基地局や無線 LAN のアクセスポイントを使用した、端末のおおまかな位置情報へのアクセスを許可する
CAMERA	カメラの使用を許可する
RECEIVE_BOOT_COMPLETED	「端末が起動した」という内部的通知を受け取ることを許可する（当該アプリを、端末起動後に自動的に開始させる場合などに使用される）

[6] 個々のアプリがそれぞれ持つアプリケーションの動作に関する設定を記述するファイル。

2.3. ファイルのアクセス許可

Android OS は、Linux カーネルをベースとした OS である。ファイルに対してユーザ ID ベースのアクセス許可設定をすることにより、他のアプリから直接ファイルにアクセスできないように制限をかけることができる。

さらに、Android OS は、アプリ毎に異なるユーザ ID とグループ ID を割り当てるため、Android アプリが作成したファイルやディレクトリは、当該アプリが許可しない限り、当該アプリからのみしかアクセスできないようになっている（図 2-3）。

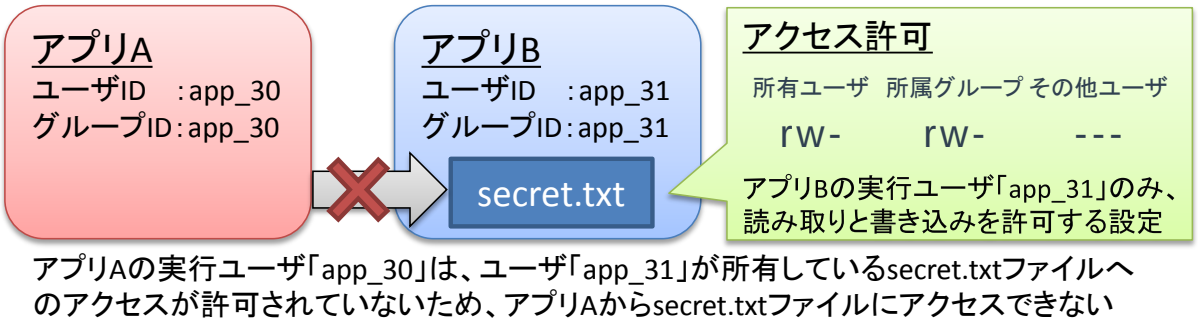


図 2-3 アクセス許可による制限例

■ デフォルトのアクセス許可設定

ファイルやディレクトリに対して、「所有ユーザ」「所属グループ」「その他のユーザ」毎に、読み取り[r]、書き込み[w]、実行[x]などのアクセス許可の設定・変更ができる。Android アプリのインストール時に生成されるディレクトリや、Android アプリが生成するファイルのデフォルトのアクセス許可設定は、以下の通りである。

[アプリインストール時に生成されるディレクトリ]

Android アプリをインストールすると、当該アプリ専用のディレクトリ /data/data/<アプリのパッケージ名>/ が生成される。生成されたディレクトリのアクセス許可は、751（所有ユーザ：rwx、所属グループ：r-x、その他のユーザ：--x）となる（図 2-4）。このアクセス許可の設定の場合、他のアプリはディレクトリ配下への移動のみ許可されることになる。

```
# ls -l -d /data/data/jp.go.ipa.fs.permissions
ls -l -d /data/data/jp.go.ipa.fs.permissions
drwxr-x--x app_41   app_41          2012-05-31 04:40 jp.go.ipa.fs.permissions
```

図 2-4 アプリ専用ディレクトリのアクセス許可設定（Windows XP の cmd.exe から確認）

[アプリが生成するファイル]

Android アプリが作成するファイルのアクセス許可は、660（所有ユーザ：rw、所属グループ：rw、その他のユーザ：---）となる（図 2-5）。このアクセス許可の設定の場合、他のアプリから当該ファイルの読み書きはできない[7]。ただし、Android アプリがファイル作成時の設定を明示的に変更することにより、他のアプリからの読み書きが可能となるアクセス許可を設定することもできる。

[7] Context | Android Developers
http://developer.android.com/reference/android/content/Context.html#MODE_PRIVATE 〔2012 年 5 月末日時点〕

```
# pwd
pwd
/data/data/jp.go.ipa.fs.permissions/files
# ls -l test.txt
ls -l test.txt
-rw-rw---- app_41   app_41          4 2012-05-31 04:41 test.txt
```

図 2-5 ファイルのアクセス許可設定（Windows XP の cmd.exe から確認）

■ ファイルの保存先により異なるアクセス許可設定

Android 端末には、データを保存する領域が 2 種類ある。本書では、それぞれ内部ストレージと外部ストレージと呼ぶ。内部ストレージは、Android 端末内部に搭載された記憶領域を指し、機微な情報の保存に適したストレージである。外部ストレージは、一般的には SD カードなどの取り外し可能なメディアを指すが、外部ストレージを内蔵している端末も存在する。公開してもよい情報の保存に適したストレージである。

ファイルのアクセス許可設定により、アプリ毎にアクセス制限が可能なのは、内部ストレージのみである。SD カードのような外部ストレージは、アプリ毎にアクセス制限できない（図 2-6）。外部ストレージ上のファイルは、不正なアプリを含むすべての Android アプリが読み書きできてしまう点は、注意が必要である。

```
# pwd
pwd
/mnt/sdcard
# ls -l test.txt
ls -l test.txt
----rwxr-x system   sdcard_rw       8 2012-05-31 04:41 test.txt
```

図 2-6 SD カード上のファイルのアクセス許可設定（Windows XP の cmd.exe から確認）

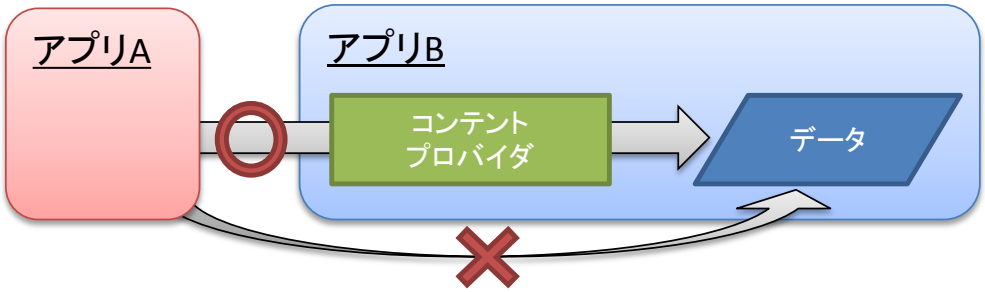
下記は、各ストレージの特徴をまとめた表である。

記憶領域	特徴
内部ストレージ	・Android 端末内部に搭載された記憶領域 ・ファイルのアクセス制限ができる ・インストールしたアプリ専用のディレクトリが作成される
外部ストレージ	・SD カードなどの取り外し可能なメディア ・取り外し不可能な内蔵ストレージとして「外部ストレージ」が用意されている端末もある ・内部ストレージと比較して記憶容量が大きい傾向にある ・ファイルのアクセス制限はできない

2.4. 他のアプリとのデータ共有

前節 2.3 の通り、Android OS は、アプリが明示的に許可しない限り、アプリ間のデータアクセスができない。図 2-7 を例に挙げると、アプリ A はアプリ B が保存しているデータに直接アクセスできないわけだ。だが、アプリ間でまったくデータを共有できないと、すべてのアプリが個別にデータを管理しないといけなくなる。そのため、Android OS は「コンテントプロバイダ」というデータ共有の仕組みを備えている。

Android アプリにおいてコンテントプロバイダを実装すると、当該アプリが管理するデータを他のアプリに公開することができる。他のアプリは、コンテントプロバイダ経由で公開されたデータへアクセスすることが可能になる（図 2-7）。



コンテントプロバイダを経由することにより、データにアクセスできる

図 2-7 コンテントプロバイダの使用イメージ

■ 他のアプリとのデータ共有の仕組み（コンテントプロバイダ）によるアクセス制限

コンテントプロバイダ経由でアクセスできる範囲を制限したり、読み取りや書き込みを制限したりすることができる。制限は、AndroidManifest.xml ファイルに記述する。

例えば、下図のようにアプリ B が「データ 1」、「データ 2」、「データ 3」を保存していた場合、「データ 1」と「データ 2」のみ他のアプリに公開して、「データ 3」は非公開にできる（図 2-8）。また、特定の権限を持つアプリのみデータの読み取りを許可する、といった細かい制限も可能だ。この「コンテントプロバイダ」の仕組みを使うことで、アクセス制限をかけた上で他のアプリにデータを公開することができる。

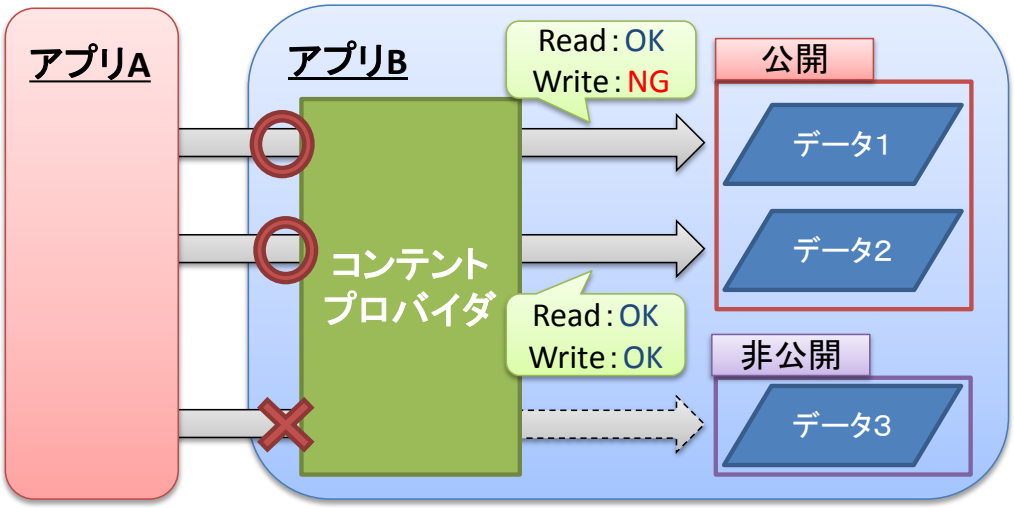


図 2-8 コンテントプロバイダのアクセス制限イメージ

2.5. 他のアプリとの機能連携

Android OS は、アプリ間の機能連携を実現するために、Android アプリから他のアプリの機能（コンポーネント）を呼び出す仕組みを提供している。この仕組みを使用することで、他のアプリと連携した Android アプリが作成できる。このアプリ間の連携のための仕組みを「インテント」と呼ぶ。例えば、インテントを使用することにより、カメラアプリで撮影した写真を、別アプリのメール送信機能を使い、友人にメール送信するといった、アプリ間の連携が容易になる（図 2-9）。

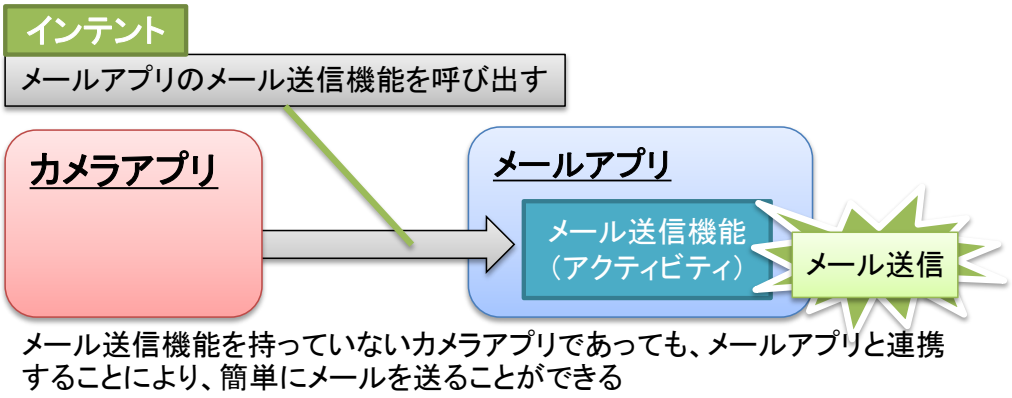


図 2-9 インテントによる機能の呼び出しイメージ

■ 他のアプリとの機能連携の仕組み（インテント）に対するアクセス制限

インテントにより、他のアプリの機能を使用することができるが、すべてのアプリに機能の使用を許可してしまうと、意図しないアプリから機能を悪用される可能性がある。呼び出しを制限することで、意図しないアプリから機能を使用されることを防げる。

制限は、呼び出される側のアプリの AndroidManifest.xml ファイルに記述する。例えば、他のアプリに機能を使わせないよう制限したり、特定の権限を持つアプリからのみ機能を呼び出せるように制限したりすることが可能だ（図 2-10）。

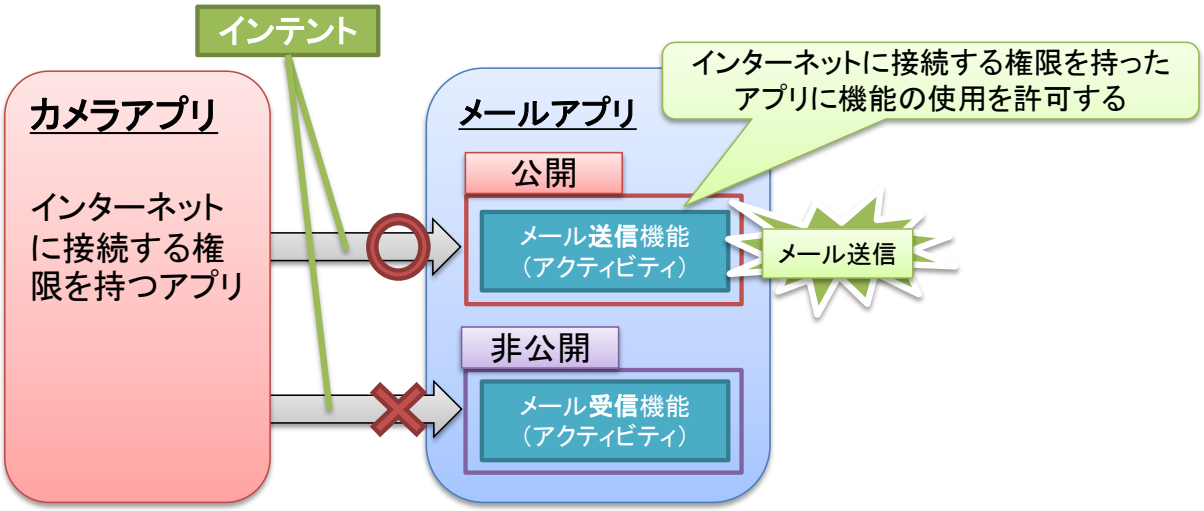


図 2-10 インテントのアクセス制限イメージ

■ **「インテント」の種類**

インテントには、呼び出したいアプリを直接指定する「明示的インテント」と、アプリは指定せずに、条件に合致する機能を持つアプリを呼び出す「暗黙的インテント」の 2 種類がある。インテントの種類により、呼び出しに関する制限の設定方法が異なる。

インテントの種類	制限の設定方法
明示的インテント	呼び出しに関する制限は、呼び出される側のアプリにおいて設定する。制限は、AndroidManifest.xml ファイルにコンポーネント毎に記述する。記述に関しては、4.2.（1）「脆弱性を作り込まないポイント」を参照いただきたい。
暗黙的インテント	条件を満たすアプリかどうかは、呼び出される側のアプリにおいて設定する。設定は、AndroidManifest.xml ファイルにコンポーネント毎に記述する。この設定を「インテントフィルタ」と呼ぶ。

3. IPA に報告された Android アプリの脆弱性

IPA は、「情報セキュリティ早期警戒パートナーシップ」において脆弱性関連情報の届出を受け付ける機関として活動している。本章では、IPA に届け出られた Android アプリの脆弱性の傾向を踏まえ、脆弱性が作り込まれた原因を考察する。

3.1. 脆弱性の傾向

2012 年 5 月末までに IPA に届け出られた Android アプリの脆弱性は累計 42 件である。これらの脆弱性の傾向を分析すると、「アクセス制限の不備」に起因するものが多いことがわかる（図 3-1）。

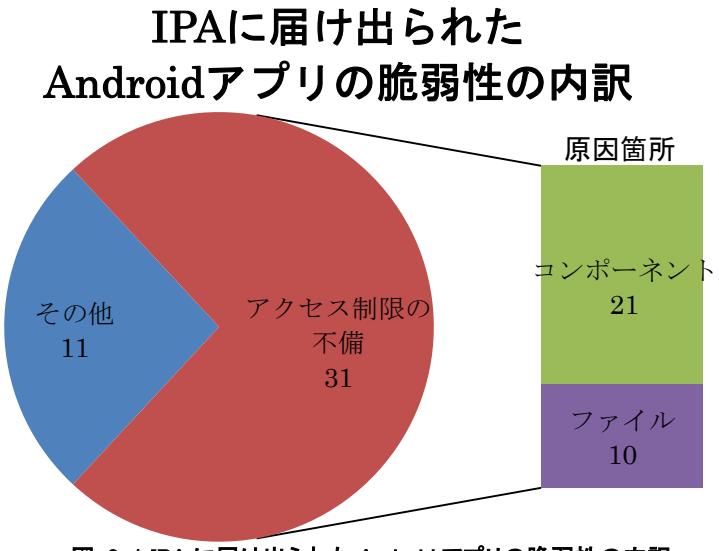


図 3-1 IPA に届け出られた Android アプリの脆弱性の内訳

さらに、「アクセス制限の不備」の脆弱性の原因箇所を分類すると、アクティビティ[8]やコンテントプロバイダなど Android アプリを構成する要素である「コンポーネント」のアクセス制限の不備と、Android アプリが生成する「ファイル」のアクセス制限の不備に大別することができる。

3.2. 脆弱性が作り込まれた原因の考察

IPA に届け出られた「アクセス制限の不備」の脆弱性は、2 章「Android の仕組み」で解説したファイル作成時のアクセス許可の設定や、アクティビティやコンテントプロバイダなどのコンポーネントのアクセス制限に関する設定不備から生じていることが分かった。これらの設定は、Android においては基本的な設定である。しかし、多くの届出があったという事実から、この<u>Android 特有の設定内容が開発者に周知できておらず、結果的に、アクセス制限の不備の脆弱性を作り込んでしまっているのではないかと IPA では推測する。</u>

次の章では、脆弱性を作り込まないポイントを交えて「アクセス制限の不備」の脆弱性例を中心に解説する。

[8] Android アプリを構成する要素の一つ。電話をかけたり、写真を撮ったり、メールを送るなどのユーザインターフェースを提供するコンポーネント。

4. 脆弱性例の紹介

本章では、2 章「Android の仕組み」で説明した仕組みを適切に使用しなかったために作り込まれた Android アプリの脆弱性を中心に紹介する。この脆弱性が悪用されると、不正なアプリから脆弱なアプリのデータにアクセスされたり、脆弱なアプリの機能を使用されたりする。

4.1. ファイルのアクセス制限不備の脆弱性

ファイルのアクセス許可の設定に不備があると、他のアプリから意図しないデータの書き換えや、読み取りをされたりする。脆弱性の例を 2 つ紹介する。

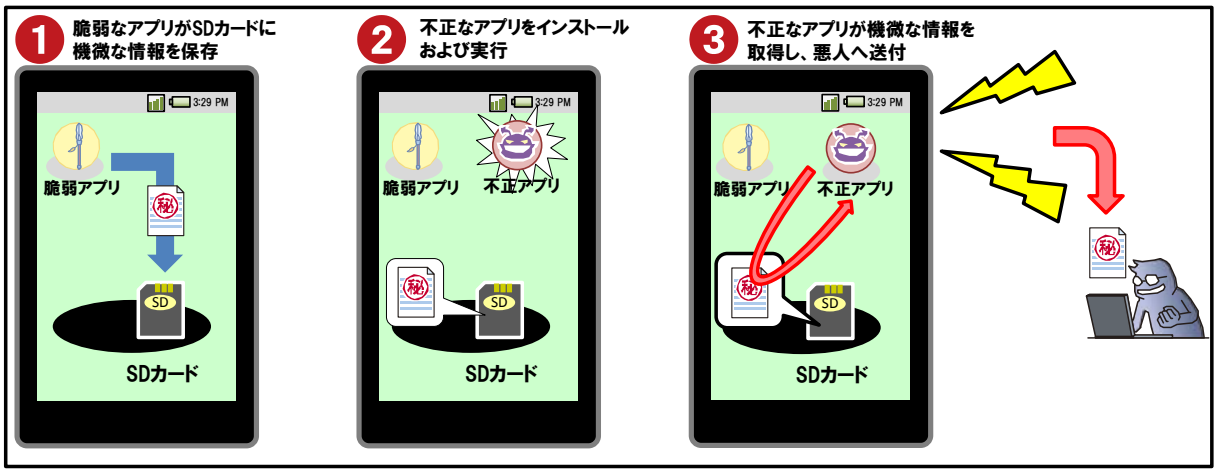
（1） SD カードに機微な情報を保存

● 脆弱性概要

SD カードにあるデータは、アクセス許可設定によるアクセス制限ができない。そのため、SD カードに機微な情報を保存した場合、他のアプリが当該情報にアクセス可能な状態となる。

● 攻撃シナリオ

1) 脆弱なアプリが機微な情報を SD カード内に保存する
2) 利用者が不正なアプリをインストールし、起動する
3) 不正なアプリが SD カード内のデータの読み取り、悪意あるサイトへ機微な情報を送信する



<u>本シナリオにおいて不正なアプリが必要とする権限</u>

・INTERNET

アプリインストール時に特に権限を要求せずとも、SD カード内のデータを読み取れるため、不正なアプリは”INTERNET”の権限のみで、SD カード内のデータを外部に送信することができる。なお、不正なアプリが SD カード内のデータを書き換える場合は、当該アプリに WRITE_EXTERNAL_STORAGE の権限が必要である。

● 脆弱性を作り込まないポイント

➢ SD カードに機微な情報を保存しない

Android 端末本体に記録できるデータ容量には限りがあるため、画像や動画など比較的サイズが大きなデータは、SD カードに保存することが一般的である。しかし、アプリから SD カード内のデータを読み取る場合、権限は不要であり、かつファイルのアクセス許可による制限もできないため、機密性の高い情報の保存には不向きである。他のアプリから読み取られてはならないデータについては、適切なアクセス許可設定を施した上で/data/data/<アプリのパッケージ名>/ 配下に保存するようにする。

➢ SD カード内のデータを暗号化する

SD カードはファイルのアクセス許可による制限ができない。そのため、不正なアプリから SD カード内のデータを保護するために、データを暗号化することも有効であると考える。暗号化は、暗号鍵を用いた方法がある。暗号鍵の生成方法や管理方法の例として、暗号鍵はアプリの初期設定時に乱数で生成し、内部ストレージにあるアプリ専用ディレクトリ配下に鍵を保存する方法がある。

ただし、Android 端末の紛失や盗難により、悪意ある人に端末がわたった場合は、悪意ある人に内部ストレージにある鍵を取得されてしまう可能性がある。そのため、まずは、データを暗号化することによりどのような脅威から情報を保護したいのか、機微な情報を端末内に保存する必要があるのか等、アプリに求める要件を明確にすることが重要である。

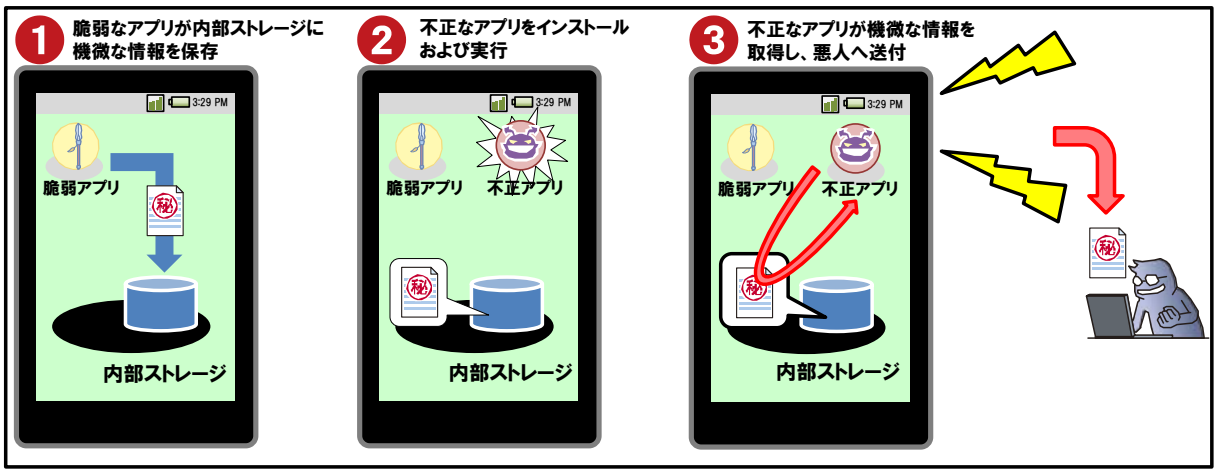
（２） ファイルが不正なアプリからアクセス可能

● 脆弱性概要

/data/data/<アプリのパッケージ名>/ 配下にあるファイルは、他のアプリからのアクセスを制限することができるが、ファイルのアクセス許可の設定に不備がある場合、他のアプリが当該ファイルにアクセス可能な状態となる。

● 攻撃シナリオ

1) 脆弱なアプリが/data/data/<アプリのパッケージ名>/ 配下に、適切なアクセス許可を設定せずに機微な情報を保存する
2) 利用者が不正なアプリをインストールし、起動する
3) 不正なアプリが /data/data/<アプリのパッケージ名>/ 配下のデータの読み取り、悪意あるサイトへ機微な情報を送信する



本シナリオにおいて不正なアプリが必要とする権限

・INTERNET

●脆弱性を作り込まないポイント

➢ ファイル作成時に、ファイルへのアクセス許可を適切に設定する

openFileOutput()メソッドや getSharedPreferences()メソッドを使用しファイルを作成する際は、下記のようにファイルのアクセス設定に関するモードを指定することができる。MODE_PRIVATE を指定することにより、他のアプリからのアクセスを制限することができる。

モード	説明
MODE_PRIVATE	・ファイルの所有者ユーザ（ユーザ ID が同一であるユーザ）のみアクセス可能 ・デフォルトのモード
MODE_WORLD_READABLE	・すべてのアプリが読み取りアクセス可能
MODE_WORLD_WRITEABLE	・すべてのアプリが書き込みアクセス可能

なお、他のアプリからのアクセスを許可したいが、より細かいアクセス制限をしたい場合は、コンテントプロバイダを用いたデータの公開を検討するとよい。

4.2. コンポーネントのアクセス制限不備の脆弱性

アクティビティやコンテントプロバイダなどの Android アプリを構成する要素である「コンポーネント」にアクセス制限の不備があると、インテントを使用し、他のアプリから意図せず機能を使用されたり、データを不正に操作されたりする。脆弱性の例を 2 つ紹介する。

（1）不正なアプリに機能を悪用される

● 脆弱性概要

インテントにより他のアプリの機能（アクティビティ）を呼び出すことができるが、アクセス制限をしていないと、不正なアプリによりアクティビティを呼び出され、特定の機能を悪用される可能性がある。

● 攻撃シナリオ

前提

本シナリオにおける脆弱なアプリは、データをファイル共有サイトへアップロードするアクティビティを持っており、そのアクティビティにアクセス制限をかけていないものとする。

手順

1) 利用者が不正なアプリをインストールし、起動する
2) 不正なアプリは、一見動画を再生しているようにみえる
3) バックグラウンドで不正なアプリが脆弱なアプリのアクティビティを悪用して、SD カード内にあるすべてのデータをファイル共有サイトへアップロードする

本シナリオにおいて不正なアプリが必要とする権限

・なし

本シナリオの場合、不正なアプリは、INTERNET の権限がなくとも、SD カード内のデータを外部サイトにアップロードすることができる。

● 脆弱性を作り込まないポイント

➢ アクティビティやサービスに対してアクセス制限をかける

AndroidManifest.xml ファイルの<activity>要素や<service>要素に対して下記の属性を設定することで、他のアプリからの明示的インテントの呼び出しに対してアクセス制限をかけられる。

属性	説明
android:enabled	・アクティビティやサービスの有効/無効を設定する属性 ・値が true は有効、false は無効 ・デフォルト値は true
android:exported	・他のアプリからアクティビティやサービスを呼び出すことができるかどうかを設定する属性 ・値が true は呼び出し可能、false は呼び出し不可 ・false の場合は、アプリ自身、または実行ユーザ（ユーザ ID）が同一のアプリからのみ呼び出し可能 ・デフォルト値は、インテントフィルタが設定されている場合は true、されていない場合は false
android:permission	・アクティビティやサービスを呼び出す際に必要な権限を設定する属性 ・必要な権限を設定しなければ、権限なしで呼び出せる

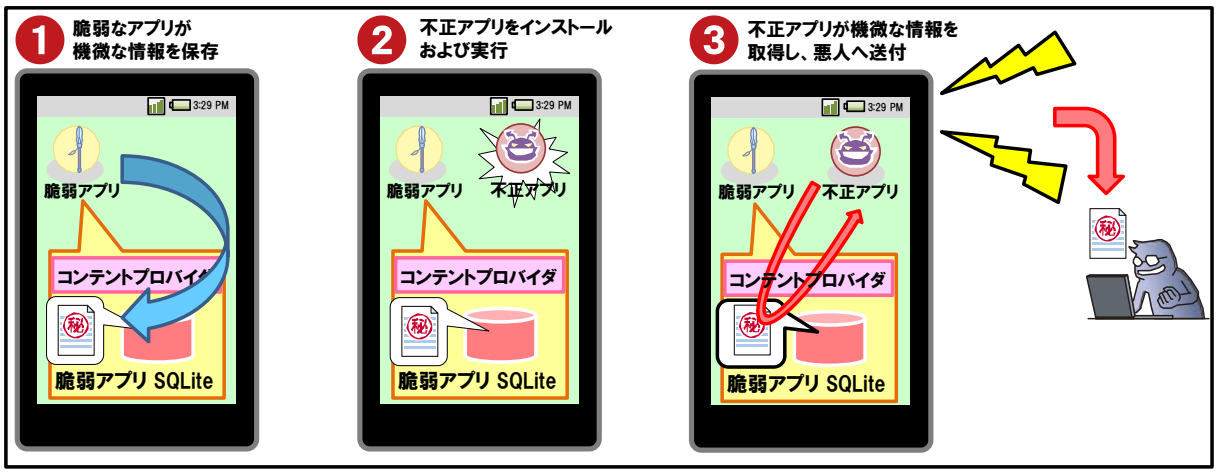
（2） ファイルが不正なアプリからアクセス可能

● 脆弱性概要

コンテントプロバイダを実装すると、特定のデータを他のアプリに公開することができるが、公開範囲や読み書きなどのアクセス制限が不適切な場合、公開したデータの読み取りや書き換えが、不正なアプリからも可能な状態になる。

● 攻撃シナリオ

1) 脆弱なアプリが/data/data/<アプリのパッケージ名>/ 配下に、機微な情報を保存する
2) 利用者が不正なアプリをインストールし、起動する
3) 不正なアプリがコンテントプロバイダを経由してデータを読み取る、または書き換える



本シナリオにおいて不正なアプリが必要とする権限

・INTERNET

●脆弱性を作り込まないポイント

➢ コンテントプロバイダに適切なアクセス制限をかける

コンテントプロバイダは他のアプリとデータの共有をする際に有用な仕組みだが、アクセス制限に不備があると、不正なアプリからデータを操作される可能性があるため、適切にアクセス制限をかける必要がある。

アクセス制限は、AndroidManifest.xml ファイルの<provider>要素に対して下記の属性で設定できる。他にも、一時的にデータを公開するために使用する android:grantUriPermissions 属性や、データの公開範囲をより細かく設定できる<path-permission>要素などが存在する。

属性	説明
android:enabled	・コンテントプロバイダの有効/無効を設定する属性 ・値が true は有効、false は無効 ・デフォルト値は true
android:exported	・他のアプリからコンテントプロバイダを呼び出すことができるかどうかを設定する属性 ・値が true は呼び出し可能、false は呼び出し不可 ・false の場合は、アプリ自身、または実行ユーザ（ユーザ ID）が同一のアプリからのみ呼び出し可能 ・デフォルト値は true
android:permission	・コンテントプロバイダを呼び出す際に必要な権限を設定する属性 ・必要な権限を有するアプリは読み取り、および書き込みを許可する
android:readPermission	・必要な権限を有するアプリに読み取りを許可する属性
android:writePermission	・必要な権限を有するアプリに書き込みを許可する属性

4.3. ログ出力に関する情報漏えい

これまで IPA への届出が多かった「アクセス制限の不備」に関する脆弱性について紹介した。「アクセス制限の不備」以外にも、件数は少ないものの Android 特有の設定に関する脆弱性の届出があった。その中から本節では、ログに機微な情報を出力しているために情報が漏えいしてしまう脆弱性を紹介する。

■ Android アプリが出力するログ情報

Android アプリにおいて、System.out.println()メソッドやandroid.util.Log クラスのメソッドの出力結果などが Android OS が管理するログに出力される。また、インテント呼び出し時のパラメータがログに出力される場合もある。

ログは logcat コマンドで確認でき、ログを読み取るための READ_LOGS 権限を持つアプリや、Android SDK がインストールされている PC から取得できる。

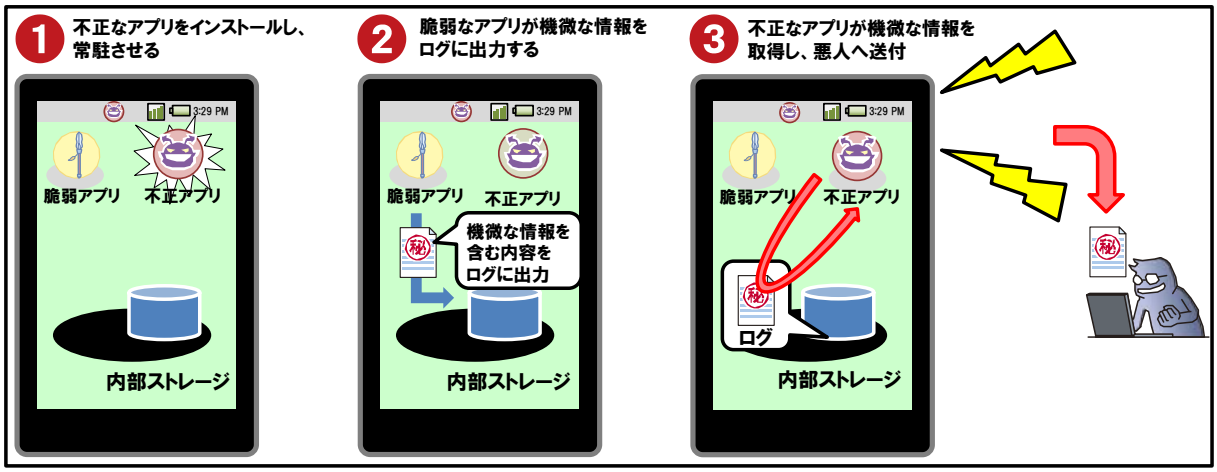
（1） 機微な情報をログに出力

● 脆弱性概要

デバッグ情報の内容はログに出力される。デバッグ情報に機微な情報を含めていると、ログからその情報が漏えいする可能性がある。

● 攻撃シナリオ

1) 利用者が不正なアプリをインストールし、常駐させる
2) 脆弱なアプリが機微な情報を含む情報をログに出力する
3) 常駐している不正なアプリが、ログに記録された 2)の情報を取得し、悪意あるサイトに送信する



本シナリオにおいて不正なアプリが必要とする権限

・INTERNET
・READ_LOGS
・RECEIVE_BOOT_COMPLETED

● 脆弱性を作り込まないポイント

➢ ログに機微な情報を出力しないようにする

ログは、READ_LOGS の権限があるアプリから閲覧できる。そのため、Android アプリを公開する際には、ログに機微な情報を出力していないか確認し、不要な情報は出力しないようにする。

例えば、デバッグ情報やインテントを送信する際のパラメータがログに出力される（図 4-1）。インテントのパラメータ情報は、明示的インテントにより同一アプリ内のアクティビティを呼び出した場合でもログに出力される。そのため、漏えいを許容できない情報は、インテントを送信する際のパラメータに含めないようにする必要がある。

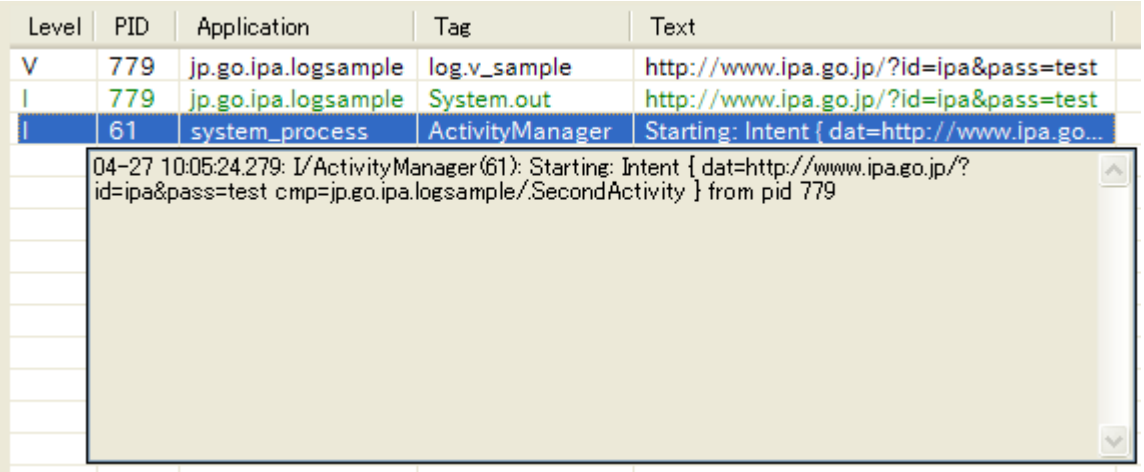


図 4-1 ログの出力例（Eclipse の LogCat ビューから確認）

上記ログ出力の該当コード

```
Intent intent = new Intent(FirstActivity.this, SecondActivity.class);
intent.setData(Uri.parse("http://www.ipa.go.jp/?id=ipa&pass=test"));
Log.v("log.v_sample", intent.getDataString());
System.out.println(intent.getDataString());
startActivity(intent);
```

5. 簡易チェックリスト

このチェックリストは、本書で紹介した各脆弱性の確認項目をリスト化したものである。自組織で開発した Android アプリに対し、対策の要/不要、対策の実施有無を記録し、セキュリティ実装の対応状況を確認する資料として活用いただければ幸いである。

■ チェック方法について

各項目の対応状況について、現状をチェックし、未対応の項目に対しては、内容欄の対応を検討してください。

☐ 対応済

対策を実施している場合に選択

☐ 未対応

対策の実施は必要であるが、何らかの理由により未実施の場合に選択

☐ 対応不要

そもそも脆弱性が存在しない実装である場合や、すでに他の対策を実施し、対策自体が不要であると判断した場合などに選択

■ Android アプリのセキュリティ実装 簡易チェックリスト

No	項目	対策内容	チェック	解説
1	ファイルのアクセス制限	○ SDカードに機微な情報を保存しない ○ 必要に応じて、SDカードに保存するデータを暗号化する	※1 ☐ 対応済 ☐ 未対応 ☐ 対応不要	4.1(1)
2		○ ファイル作成時に、ファイルへのアクセス許可を適切に設定する	☐ 対応済 ☐ 未対応 ☐ 対応不要	4.1(2)
3	コンポーネントのアクセス制限	○ アクティビティやサービスに対してアクセス制限をかける	☐ 対応済 ☐ 未対応 ☐ 対応不要	4.2(1)
4		○ コンテントプロバイダに適切なアクセス制限をかける	☐ 対応済 ☐ 未対応 ☐ 対応不要	4.2(2)
5	ログ出力の内容	○ デバッグログに機微な情報を含めない	☐ 対応済 ☐ 未対応 ☐ 対応不要	4.3(1)
6		○ インテントを送信する際のパラメータに機微な情報を含めない	☐ 対応済 ☐ 未対応 ☐ 対応不要	4.3(1)
7	アプリの権限	○ 必要以上に権限を要求しない※2	☐ 対応済 ☐ 未対応 ☐ 対応不要	2.2

※1 このチェック項目の「対応済」のチェックは、実施項目のいずれかを実施した場合にチェックする

※2 脆弱性の直接の原因ではないが、Androidの仕組みを適切に使用する上で必要な対策である

おわりに

Android OS は、「アプリケーションに必要以上の権限を持たせないよう制限するための仕組み」と「データ共有や機能連携するための仕組み」が備わっている。本書では、このような Android OS が用意している仕組みを適切に使用していないために起こる問題に焦点を当て、解説した。

現在は、Android OS 自体の脆弱性を狙ったウイルス[9]や、利用者を騙して直接情報を窃取するアプリ[10]は観測されていても、個別のアプリの脆弱性を狙った動きは少なくとも国内においては観測されていない。しかし、Android を取り巻く環境の変化は非常に早く、OS のアップグレードや公式マーケットへのマルウェア検知システムの導入によるセキュリティ向上が図られている。

そのような状況において、攻撃のトレンドも日々変化していると考えられる。有名アプリになれば利用者が数百万となるアプリも珍しくないため、アプリの脆弱性を狙う攻撃が広まる可能性は否定できない。アプリの脆弱性を狙った不正なアプリは、直接情報窃取するアプリと比較してアプリのインストール時の権限チェックで見つけることは困難である。しかし、大半が、アプリが Android の仕組みを適切に使用することにより防げる問題である。そのため、本書で取り上げた脆弱性の存在を Android アプリ提供者が認識し、提供しているアプリに脆弱性が作り込まれないように対策することを IPA は期待する。

今後も IPA は、Android アプリの脆弱性に限らず、ウェブサイト運営者やウェブアプリケーション/ソフトウェア開発者に対してセキュリティに関する有用な情報を発信できるよう活動を続けていく。

---

[9] 情報処理推進機構：IPA テクニカルウォッチ：『スマートフォンへの脅威と対策』に関するレポート
http://www.ipa.go.jp/about/technicalwatch/20110622.html 〔2012 年 5 月末日時点〕

[10] 情報処理推進機構：情報セキュリティ：ウイルス・不正アクセス届出状況について（2012 年 4 月分）
http://www.ipa.go.jp/security/txt/2012/05outline.html#5 〔2012 年 5 月末日時点〕