PacSec 2011 Tokyo

How Security Broken?

Android の内部構造とマルウェア感染の可能性

Fourteenforty Research Institute, Inc.
株式会社 フォティーンフォティ技術研究所
http://www.fourteenforty.jp

背景: Android と脅威

- シェアとともに増大するマルウェア
  - 2010 年には前年の 4 倍に増加(1)
  - 2010年8月：SMS マルウェアの発見 (FakePlayer.A)
  - 2011年3月：“削除できない” マルウェアの発見 (DroidDream)
- 脆弱性と脆弱性攻撃
  - 2003～：脆弱性攻撃を防止する実装の登場 (DEP, ASLR など)
  - モバイルデバイスへの攻撃
    - 2007～：JailbreakMe (iOS 用の攻撃コード)
    - 2011年3月：DroidDream (2 個の root 化脆弱性を利用)
- 対抗策：アンチウイルスソフトウェアの登場
  - PC と同様に守る必要がある端末に

(1) http://www.adaptivemobile.com/

アジェンダ

- 低レイヤーのセキュリティ
  - カーネルレベルでの保護機構
- Android アプリケーション レイヤーの仕組み
  - パッケージ / パーミッション
  - インテント / アクティビティ / ブロードキャスト
- 脅威とそれへの対抗策
  - マルウェアの感染と脅威
  - *root 化*の問題
  - ウイルス対策ソフトとその問題

Linux カーネルのメモリ保護機構と Android における現状

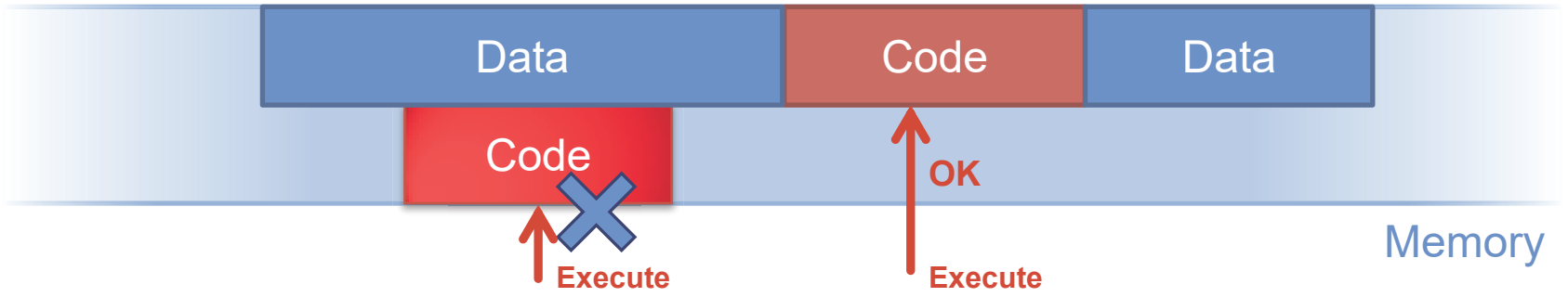
低レイヤーのセキュリティ

低レイヤーのセキュリティ実装

	-2.2	2.3-,3.0-	4.0-	iOS
DEP (スタック)	× (1)	○(1)	○	対応: 2.0-
DEP (その他)	× (2)	○	○	
ASLR (スタック)	○	○	○	対応: 4.3-
ASLR (ヒープ)	×	×	？ / × (3)	
ASLR (モジュール)	×	×	○ / × (3)	部分的対応: 4.3-(4)

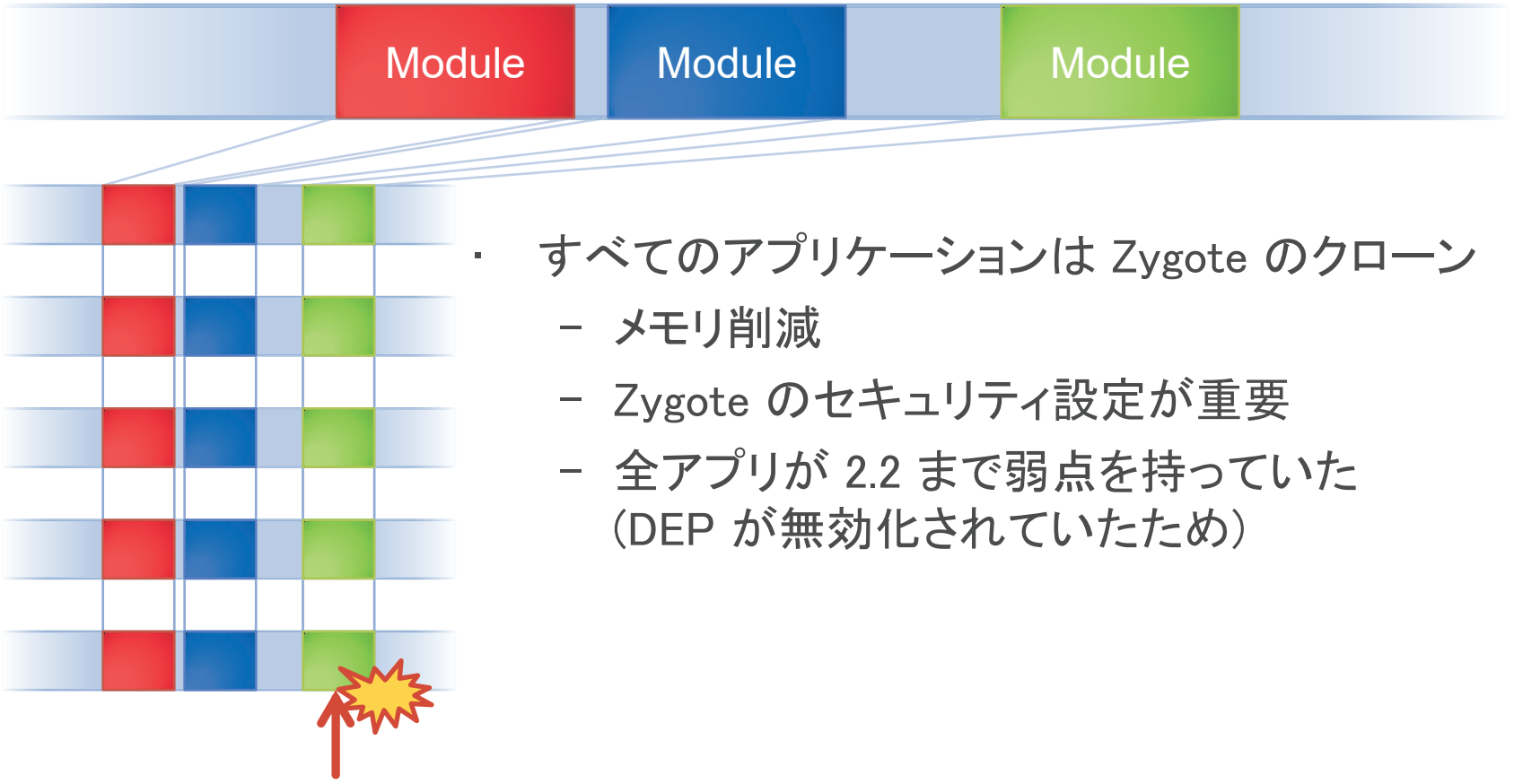
(1) ネイティブアプリケーションの場合、ビルド環境のコンパイラフラグに依存する。ここではデフォルト設定の場合を示す。
(2) 移植性のある方法でメモリ確保をした場合
(3) 前者はリリースノートからの推測 / 後者は Android SDK 付属のエミュレータにおける調査結果
(4) アプリケーションが ASLR に対応している場合のみ

セキュリティ機能：DEP



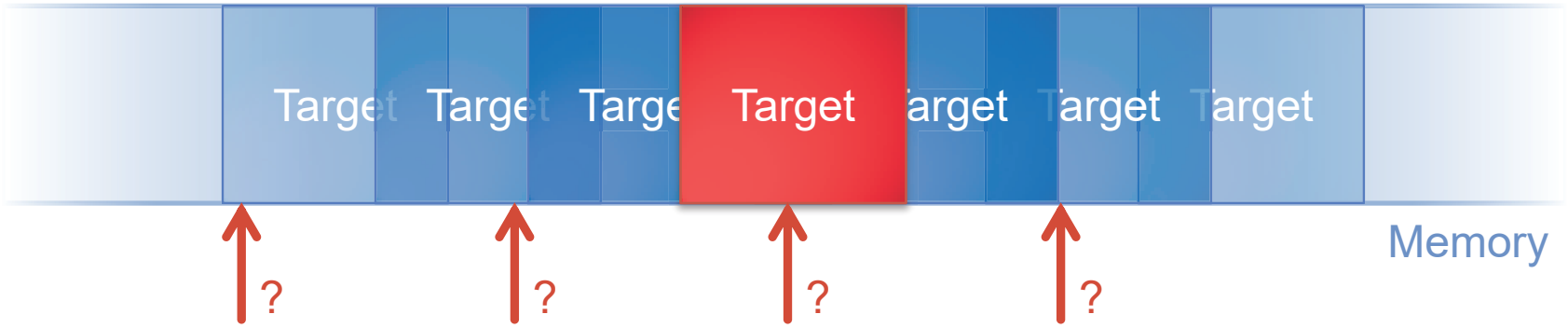
- ハードウェア的にコード（実行可能）とデータ（実行不能）領域を区別し、“データ”の実行を防止する
- DEP 有効化のためにはコンパイラフラグの指定が必要
  - Android 2.2 までは指定されていなかった
  - DEP が互換性のために無効化されてしまう
- Android 2.3 で解決

Android 内部の仕組み：Zygote



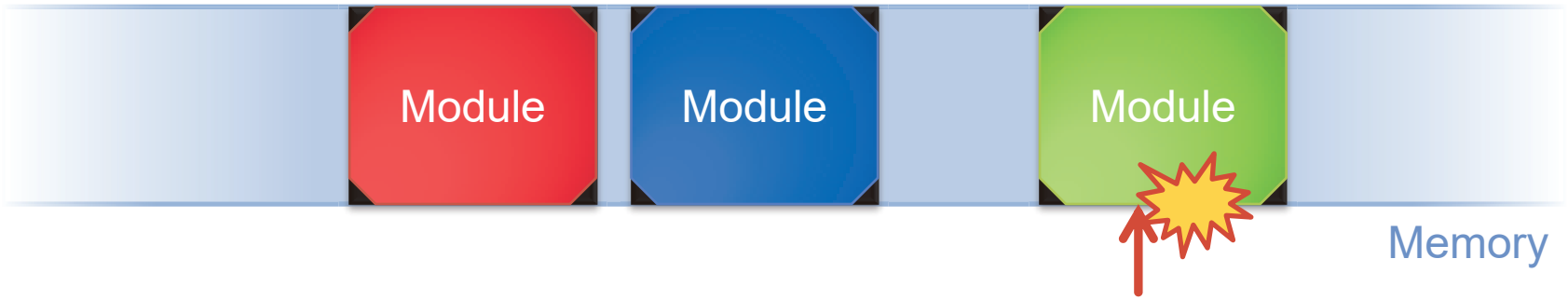
- すべてのアプリケーションは Zygote のクローン
  - メモリ削減
  - Zygote のセキュリティ設定が重要
  - 全アプリが 2.2 まで弱点を持っていた
    (DEP が無効化されていたため)

セキュリティ機能：ASLR



- メモリの配置をランダム化して攻撃を防止
  - 近年 exploit の多くは特定のアドレスを狙う
- カーネル設定：ヒープ以外すべてをランダム化（OK）
  - 実際にはモジュール（ライブラリ）はランダム化されない
  - 後述する Prelinking が働いているため

セキュリティ上の懸念：Prelinking



- Prelinking は、ライブラリなどを（コンパイル時に設定された）固定のアドレスに配置してしまう
  - メモリ削減、起動高速化のためだが…
  - ライブラリなどに対する ASLR の事実上の無効化
  - return-into-libc や ROP などの、洗練された、特定アドレスを狙う攻撃が極めて容易になってしまう
- 本来 Linux カーネルが持つセキュリティ機能を半減してしまう。

ASLR in Android 4.0?

- 2011年10月末現在、Android 4.0 搭載の端末は未出荷
  - 新しい Android SDK に Android 4.0 “Ice Cream Sandwich” のエミュレータ用イメージが追加
- Google は 4.0 において ASLR が追加されることをアナウンスした(1)
  - しかしながら、公開されているエミュレータイメージを解析する限り、今のところ ASLR は有効化されていない
  - “まともな” ASLR が実装されていることを期待する!

(1) http://developer.android.com/sdk/android-4.0-highlights.html

まとめ

- Linux カーネルが持つセキュリティ機能は、
  最新の Android においても十分に機能しているとはいえない
  - ネイティブコード部分が狙われた場合、
    高い確率で攻撃に成功してしまう危険性がある
- 不適切な実装やビルド設定は解決の余地がある
  - Android 2.3 で改善された
- Android 独自のメモリ削減の仕組みは
  メモリ保護の仕組みを大幅に弱体化させる
  - モバイル CPU の性能向上によって改善していく余地がある
  - 改善していく余地はあった! (Android 4.0)

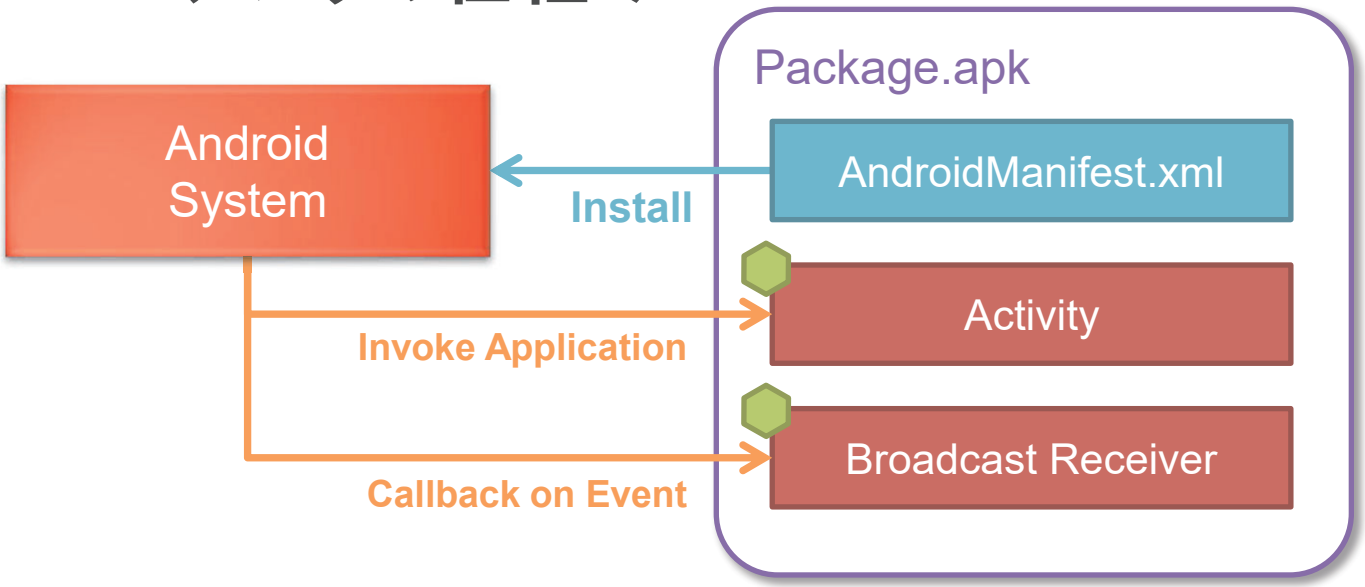
Android 独自の機構はどのように動作するか

アプリケーションレイヤー独自の仕組み

Android アプリケーションの仕組みとは?

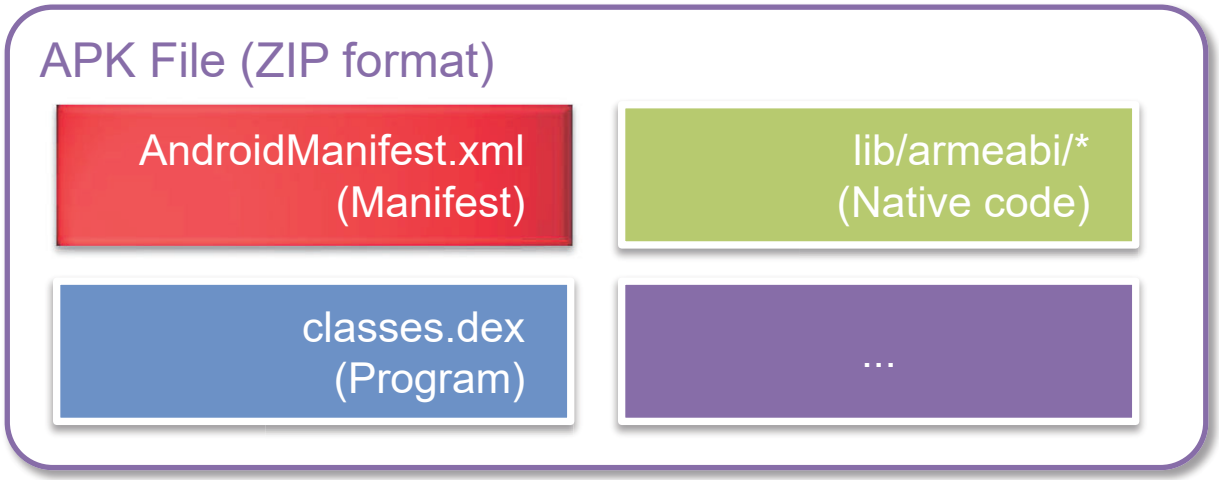
- 一般の OS におけるアプリケーションの位置づけとは全く異なる
  - インテントを利用したアプリケーション (アプリ) 同士の連携が可能
- Android セキュリティを知るには、
  これら独自の仕組みをある程度理解しておく必要がある
  - パッケージとマニフェスト
  - パーミッション
  - インテント機構
    - アクティビティ (Activity)
    - ブロードキャスト (Broadcast)
    - ...

Android : アプリの仕組み



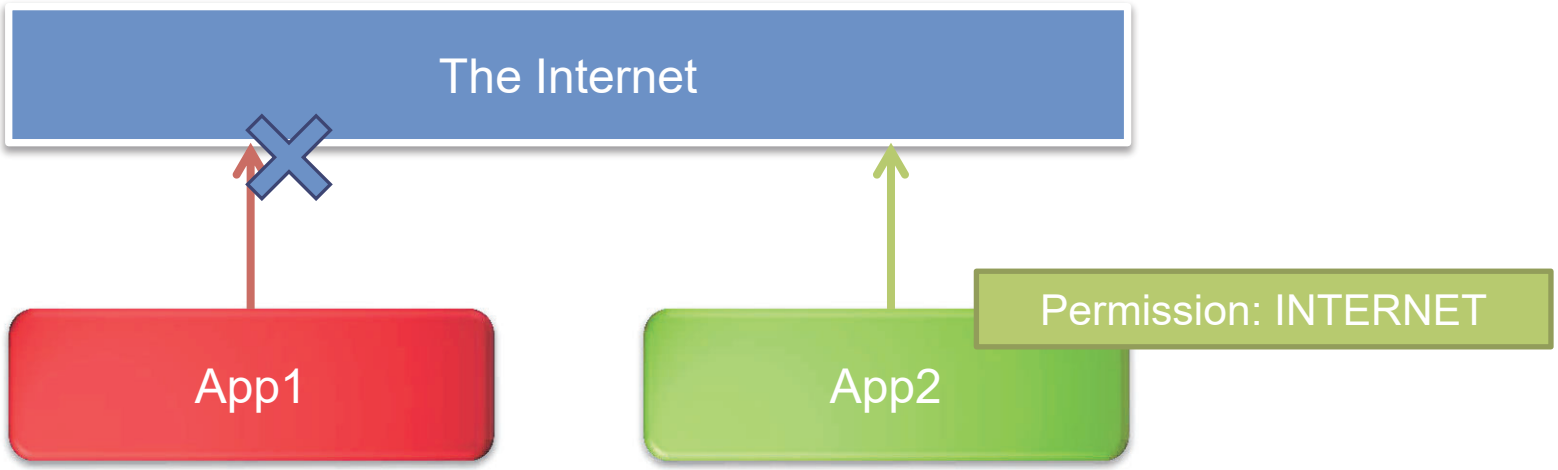
- アプリは “パッケージ” に格納される
- パッケージ内のクラスが “どのように呼び出されるべきか” を
  あらかじめ登録しておき、一定のルールでシステムから呼び出される
  - アクティビティ (Activity)
  - ブロードキャスト (Broadcast)
  - …

Android : パッケージ



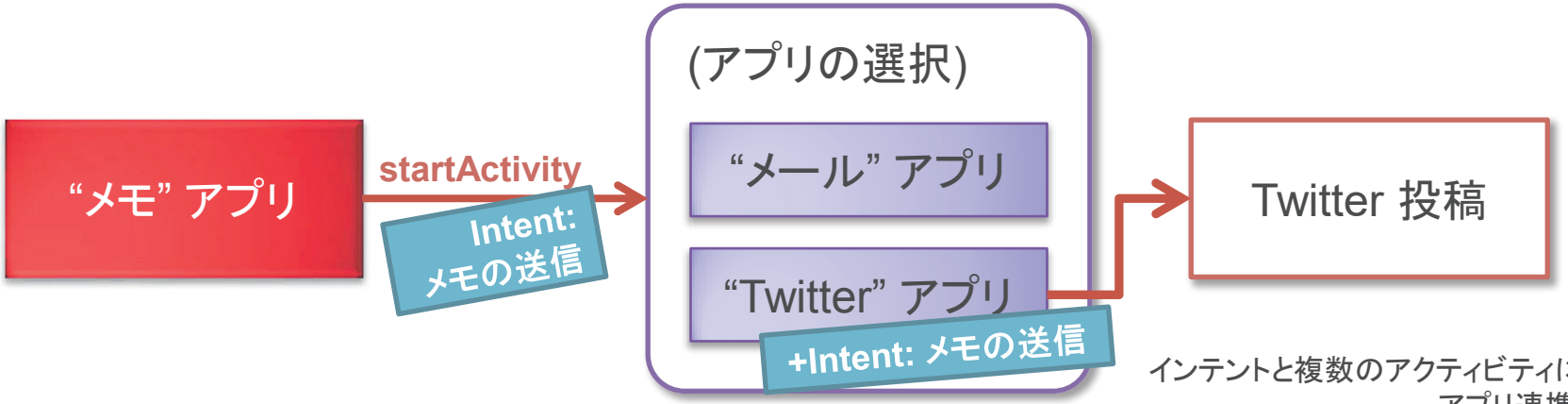
- パッケージ自体は単なる ZIP アーカイブ
- AndroidManifest.xml (マニフェスト)
  - Android アプリの情報や権限
  - クラスがどのように呼び出されるか (Activity, BroadcastReceiver...)
- インストール後には変更できない
  - ただしパッケージのアップデートで変更可能

Android : パッケージ/パーミッション



- アプリの動作に必要な「権限」をまとめたもの
  - INTERNET (インターネット接続), READ_PHONE_STATE (電話番号などの読み取り) など、100 個以上が存在する
- 必要なパーミッションがない操作は拒否される
  - 逆に言えばパーミッションさえあれば、ほとんど何でもできる

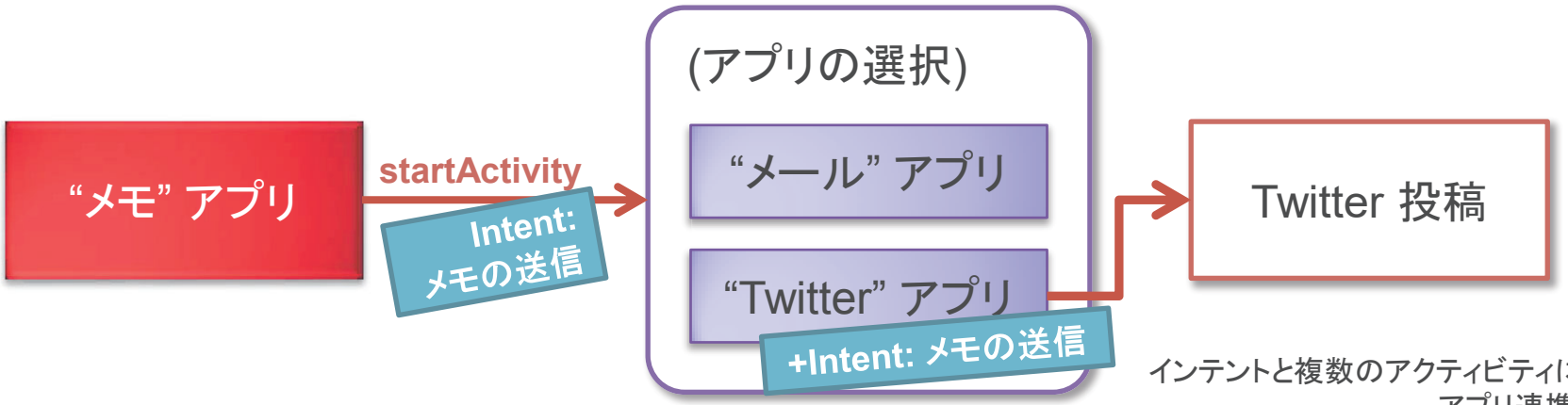
Android : インテント



インテントと複数のアクティビティによる
アプリ連携の例

- インテントと呼ばれる機構を用いてアプリ間での連携を行う
  - 操作内容、対象を含んだメッセージを送受信
- インテント機構は様々な用途に用いられ、極めて重要な位置を占める
  - アプリの相互呼び出し
  - システムイベントの通知、受信

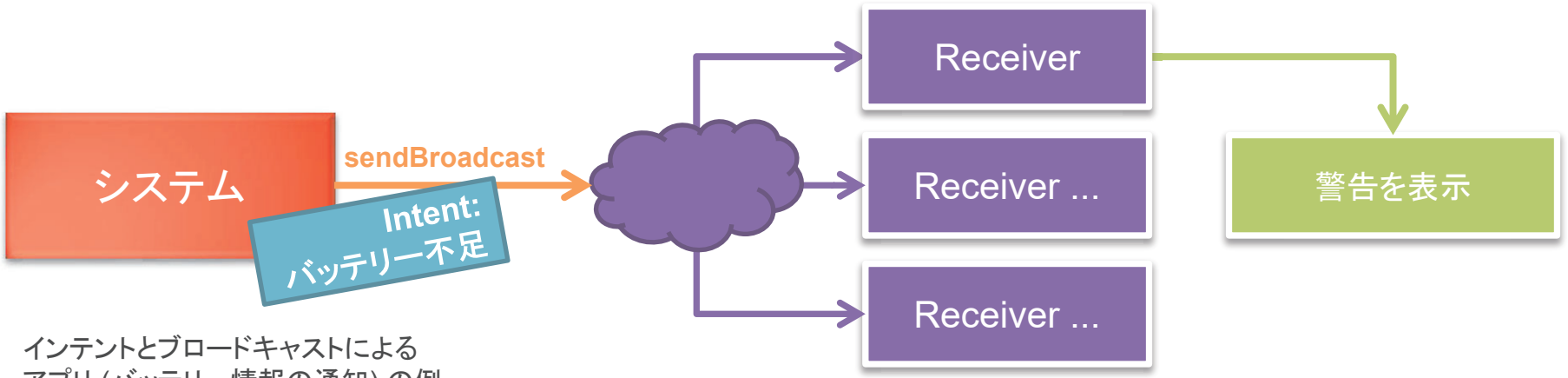
Android : インテント（アクティビティ）



インテントと複数のアクティビティによる
アプリ連携の例

- アクティビティ（Activity）は、何かを “操作” するための UI の単位
  - ここでは、“メモを作成する” “Twitter に投稿する” ための
    インタフェースがアクティビティに相当する
  - 行う操作と対象をマニフェストに記述しておけば、
    自動的にアプリ同士の連携を取ることができる
    （後述の インテント フィルタ による）

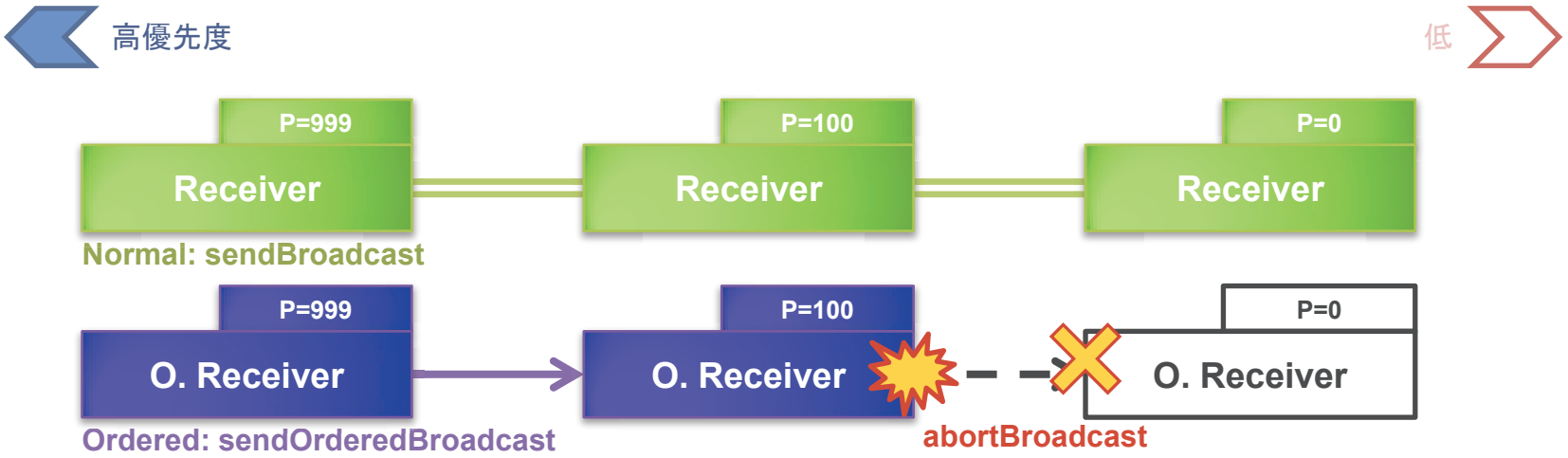
Android : インテント (ブロードキャスト)



インテントとブロードキャストによる
アプリ (バッテリー情報の通知) の例

- ブロードキャスト (Broadcast) は、システムやアプリが
  発生させたイベントを受信するための仕組み
  - 登録された (原則として) すべての ブロードキャスト レシーバ
    (BroadcastReceiver) が起動され、そこでイベントが処理される

Android : 順序付きのブロードキャスト

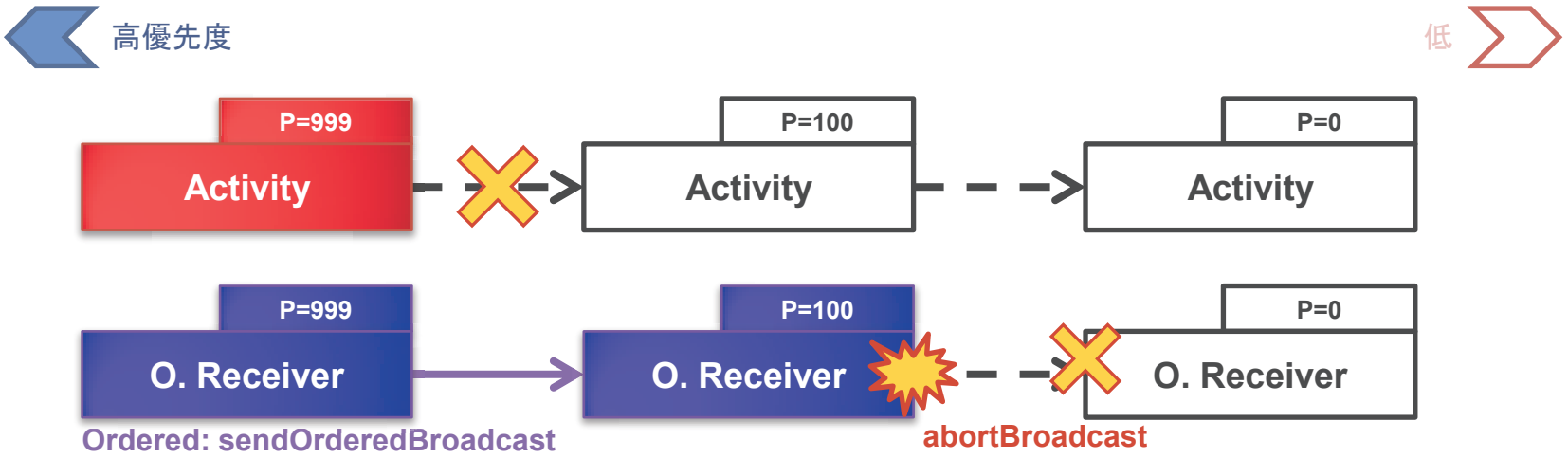


- ブロードキャストには、“順序なし” と “順序付き” のものがある
  - システム中では、一部のイベントだけが “順序付き”
- 順序付きのブロードキャストには、次の性質がある
  - 後述する優先度の順番に呼び出される（同じ優先度なら順不同）
  - abortBroadcast という操作でブロードキャストの処理を途中で中断することができる

Android : インテント フィルタ

- Windows に例えると関連付けの仕組みに近い
  - アクション (何をする/受け取るか), カテゴリ (どう実行するか)
  - ファイルの種類 (MIME タイプ), 場所, プロトコル…
- マニフェスト (AndroidManifest.xml) に記述しておく
  - システムがすべてのインテント フィルタを管理する

Android : インテント フィルタの優先度



- インテント フィルタには、その優先度を設定できる
  - 高優先度のインテント フィルタに対応するアクティビティや ブロードキャスト レシーバが優先される
  - “順序付きの” ブロードキャスト

まとめ：解説した項目

- Android アプリ
  - パッケージ / マニフェスト
  - パーミッション
- インテントを使う Android 独自の方式
  - アクティビティ（Activity）
  - ブロードキャスト（Broadcast）
    - 順序付きのもの
    - 順序付きでないもの
- インテントを補助するインテント フィルタ
  - その自由度と、優先度の仕組み

Android マルウェアや保護の現状と問題

脅威の現状、対抗策の限界

Android セキュリティと脅威の現状

- 既に、多数のマルウェアと幾つかのウイルス対策ソフトが存在
  - マルウェアの現状や脅威はどのようなものか
  - ウイルス対策ソフトで守れるのか
- マルウェア
  - 現在のトレンドと、動作の特徴
- ウイルス対策ソフトとその動作
  - 権限不足という致命的な問題
- root 化の問題
  - Android セキュリティとの関連性
  - 対抗策と、それでも解決しない問題

Android マルウェアの歴史：2009年

- 1月13日に発見（McAfee）
  - CallAccepter, Radiocutter, SilentMutter
  - root 化された Android 1.0 が対象
  - 端末に対して DoS 攻撃を引き起こすのみ
- 10月26日にリリース（公式情報による）: Mobile Spy
  - 有償のスパイウェア（SMS, GPS, 通話記録を監視する）
  - 日本で問題になった“カレログ”と様々な点で類似している

- これらは主流のサイバー犯罪との関連性は薄く、
  以降のマルウェアとは性質が明らかに異なる。

Android マルウェアの歴史：2010年

- 8月10日発見（Symantec）：FakePlayer.A
  - 最初の“本格的な” Android マルウェア
  - ロシアのウェブサイトにおいて、
    動画プレイヤーと偽って頒布されていたもの
  - プレミアム SMS を送信し、直接的に利益を生もうとする

- 現代のサイバー犯罪と Android の合流
  - これ以降、急速にマルウェアの手口が明らかに悪質化

Android マルウェアの歴史：2011年

- 1月：“再パッケージされた” Android アプリ
  - 既存のアプリケーションにマルウェアの機能を追加し、再び Android パッケージの形態に戻して頒布する
- 3月：root 化を悪用する Android マルウェア
  - システム領域に入り込み、削除を不可能にする
- 6月：外部からコードをダウンロードする Android マルウェア
  - DexClassLoader を利用したコードの動的実行
- 7月,10月：アプリのアップデート機構を悪用するマルウェア
  - 最初にダウンロードするアプリケーションには悪性コードが含まれていないものの、アップデート後にそれらの機能が追加される

マルウェア：特徴

- 主なマルウェアは次のいずれかに分類できる
  - スパイウェア
  - バックドア
  - プレミアムサービスを悪用するマルウェア（後述）
- 中国やロシアを示すマルウェアが多い
  - 特定の国を示す APN や電話番号
  - アプリケーション中に含まれる文字列
  - ただし、これらの国でしか動作しないマルウェアも
- 外部への / 外部からの情報のやりとり
  - HTTP
  - SMS

マルウェア：特徴（プレミアムサービス）

- 有料の SMS や電話サービス
  - 日本の場合：“ダイヤル Q2”
  - 有料サービスに国境はない
    （ただし、日本においてはプレミアム SMS は使用できない）
- プレミアムサービスの悪用：“ダイアラー（dialer）”
  - プレミアムサービスに意図的に接続することで、
    攻撃者が（直接）利益を得る
  - 日本では下火となっていた
  - Android スマートフォンは電話回線とつながっている!

マルウェア：インテント フィルタの使用

- 情報を盗み出したり、あるいはシステムに
  留まり続けるためにブロードキャストを受信する
  - 39/44 マルウェア検体
- SMS の受信は、“順序付き” のブロードキャストである
  - 高優先度のレシーバで abortBroadcast を使用すると、
    標準の SMS アプリでは受信したメッセージが見えなくなる
  - SMS として受信したコマンドを隠す
  - 14/44 マルウェア検体

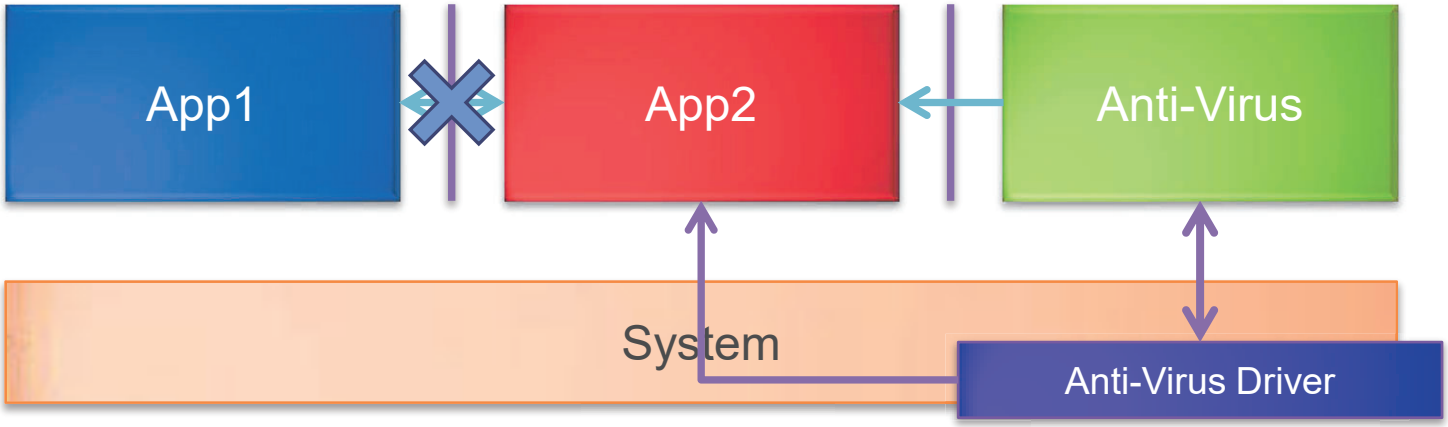
マルウェア：進化の現状

- 本格的な難読化が施されている Android マルウェアは
  現在のところ確認されていない
  - 現状のマルウェアはそれほど洗練されていない（比較的解析が容易）
- ただし、技術的には急激に進歩しつつある
  - DroidDream
    root 化を行った上で定期的にパッケージ（APK ファイル）を
    ダウンロード、自動でインストールする
  - Plankton
    外部から DEX ファイル（Dalvik バイトコード）をダウンロードし、
    クラスローダーという Java の仕組みを悪用して動的に実行する
- root 化を利用するマルウェアをはじめ、
  洗練されたマルウェアが問題になっていくと思われる

ウイルス対策ソフト：現状

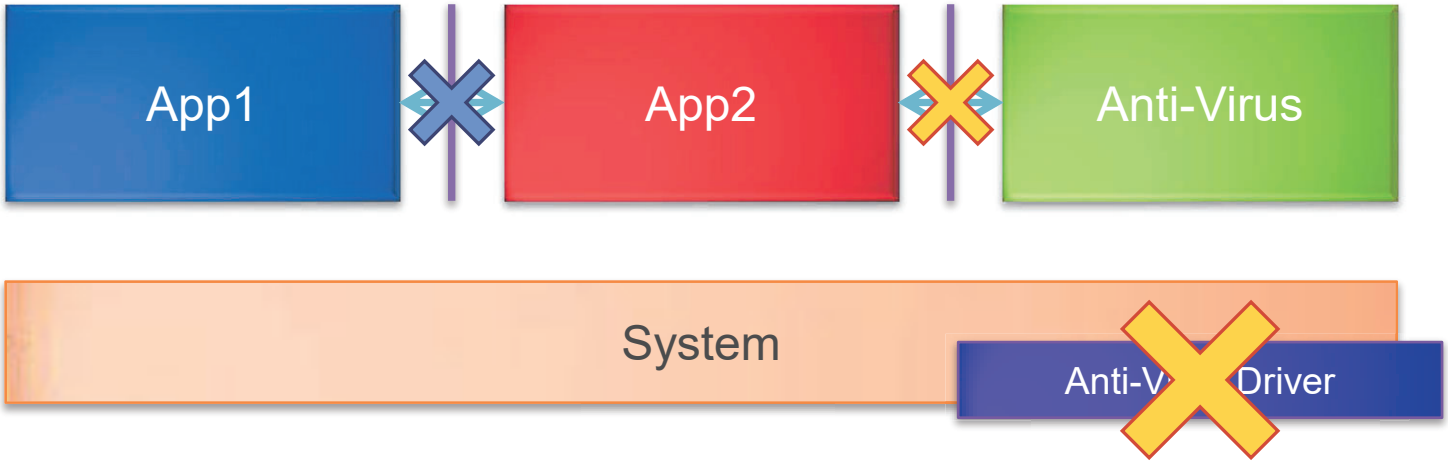
- 大量のインテントフィルタやブロードキャストの利用
  - 部分的なリアルタイム保護
  - ダウンロードしたファイルやアプリのスキャン
  - SMS や E-メールのスキャン

ウイルス対策ソフト：現状と問題



- Android 用のウイルス対策ソフトは、ユーザー権限で動いている
  - PC 向けであればシステム権限でドライバを動作させ、システムすべてを監視することができる

ウイルス対策ソフト：現状と問題



- Android というサンドボックス
  - プロセス間の不正な干渉を阻止する
  - しかし、アンチウイルスソフトウェアによる「正当な」干渉までサンドボックスに阻止されてしまう
  - もちろん、ドライバはインストールできない

ウイルス対策ソフト : その他の問題

- 検体の収集状況にばらつきがある
  - 大規模な検体収集ソフトウェアなどが
    まだまだ定着していない面がある
  - Android Market の規約ではクローラー等の使用は禁止

ウイルス対策ソフト：対等な権限

- マルウェアとウイルス対策ソフトは、同等の権限を持っている
  - 互いに互いを停止させることができる
- 動的なヒューリスティック検出は難しく、部分的
  - シグニチャに頼る必要がある
  - 保護できるのは限定的である
    - ただしここまでなら、既存のマルウェアを検出し、削除を促すことまでは可能である
- もしマルウェアが高い権限を獲得できるとしたら…
  - 管理者権限の奪取 = root 化

root 化

- 本来許可されていない端末の管理者権限を得る行為
  - 特にここでは端末のローカル脆弱性を突くもの
- root 化に使用される（Android 全般の）脆弱性
  - CVE-2009-1185（exploid）
  - [CVE 番号なし]（rage against the cage）
  - CVE-2011-1149（psneuter）
  - CVE-2011-1823（Gingerbreak）
  - [CVE 番号なし]（zergRush）
- ベンダーやチップ固有の脆弱性も複数存在する

root 化：脆弱性 (1)

- suid プログラムのロジックエラー
  - ある Android タブレットの場合: OS コマンドインジェクション

[非公開]

root 権限で任意のコマンドを実行することができる。

root 化：脆弱性 (2)

- ドライバにおける、ユーザーバッファの不適切な取り扱い
  - ある Android スマートフォンの場合: センサードライバ

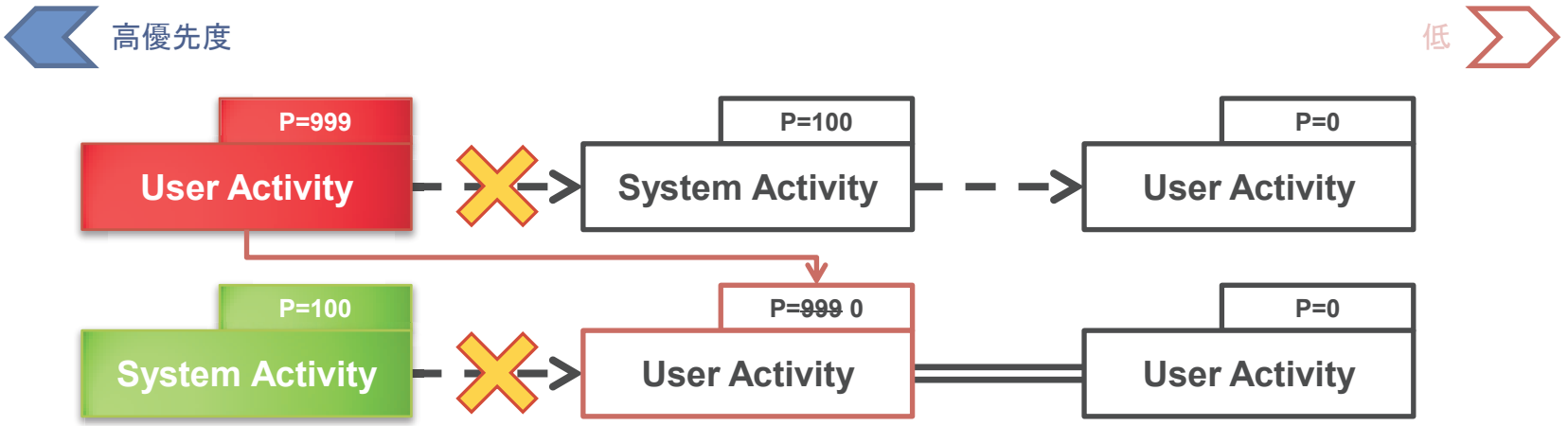
[非公開]

任意のユーザーメモリに、copy-on-write を無視して [非公開] を書き込むことができる。
この脆弱性を用いて libc を部分的に破壊することで、root 権限を得られる。

root 化：問題

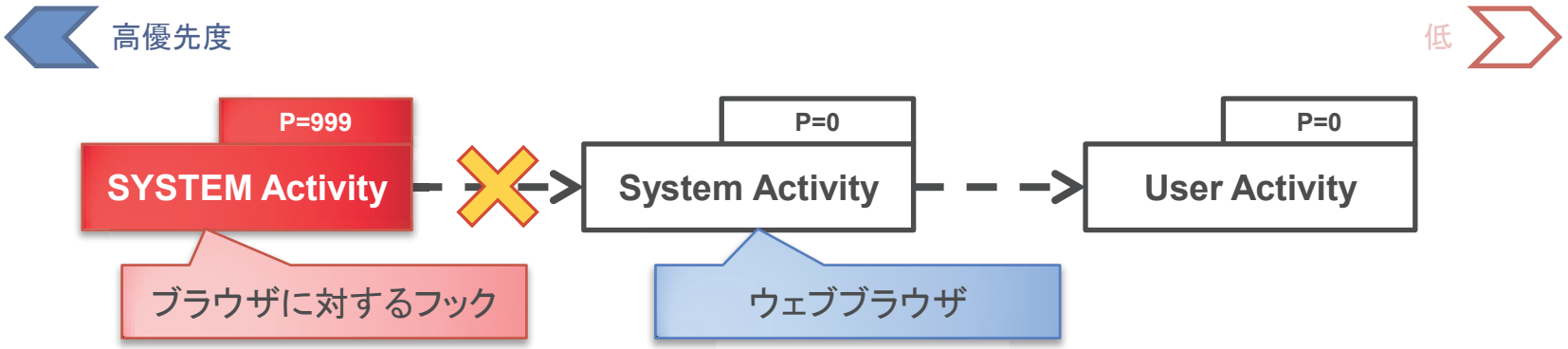
- マルウェアもほぼ同じ脆弱性を突くことができてしまう
  - マルウェアがウイルス対策ソフトより
    はるかに高い権限を獲得することが可能になる
  - ウイルス対策ソフトから逃れることができる
- また root 化によって、Android が持つ保護の仕組みが破れてしまう
  - アクティビティに対応するインテント フィルタの優先度
  - パーミッションの仕組みの保護
- これらがマルウェアによって実行されると、ウイルス対策ソフトが
  対抗できなくなってしまう可能性がある

破れる保護：Activity の優先度 (1)



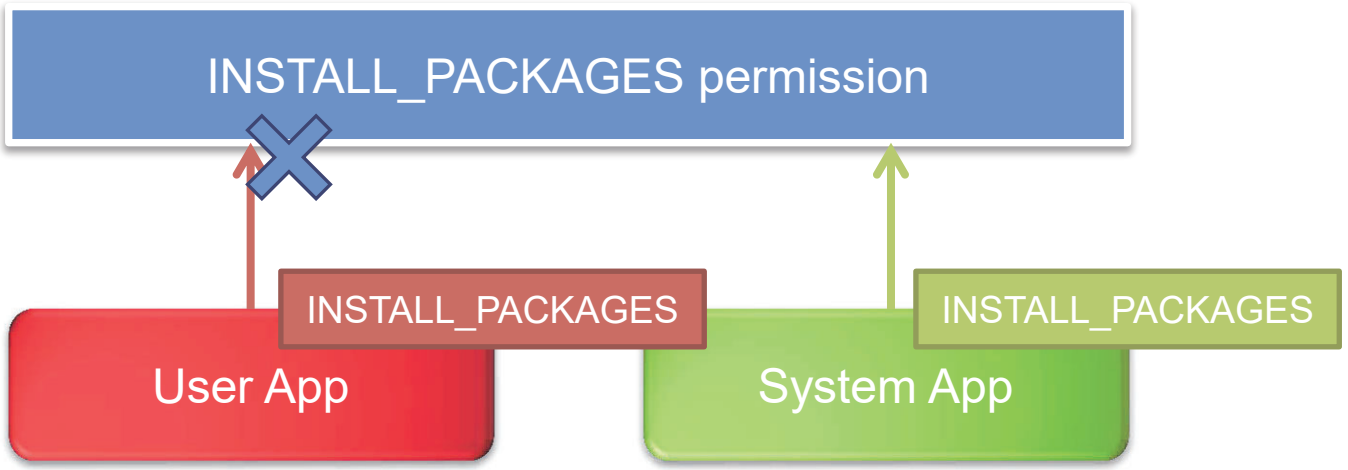
- Activity に対しても優先度を設定できるが、
  通常インストールするアプリケーションは高優先度が無効化される
  - Activity に対する高優先度は、実質的に
    Intent のフックを実現することになってしまうため
  - システムアプリケーションのみに予約されている

破れる保護：Activity の優先度 (2)



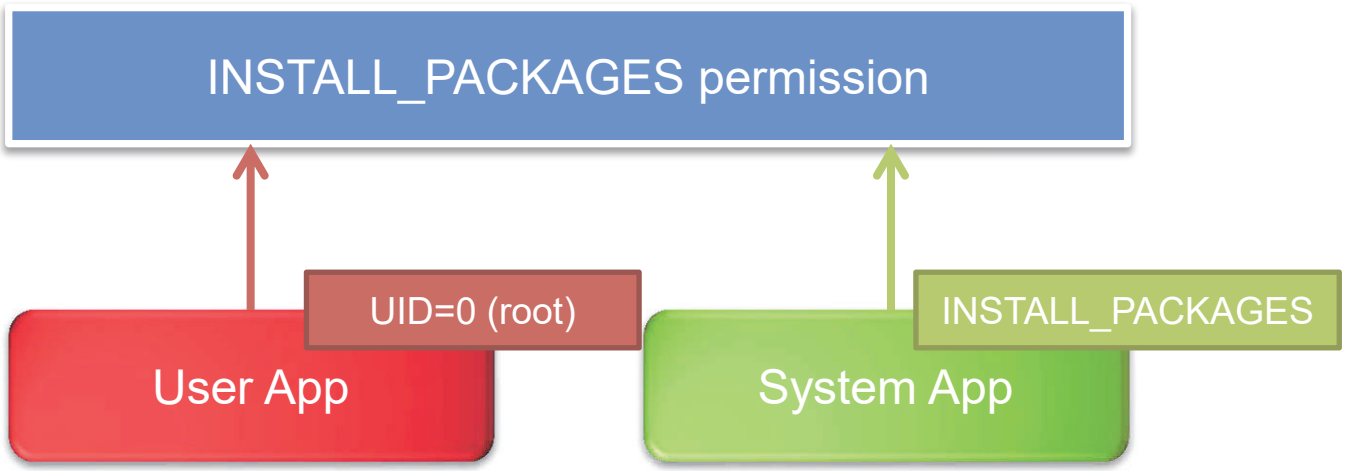
- システム領域にインストールされたパッケージの Activity は、高優先度で起動することができる
  - root 化を悪用してシステム領域にパッケージをインストールすれば、暗黙的 Intent を実質的にフックすることができる。
  - 例: ウェブブラウザの用いる Intent の一部をフックすることで、フィッシングを行うことができる
    - ただし、Android 3.0− では（仕様変更のため）動作しなくなった

破れる保護：パーミッション（1）



- システムの重要なパーミッションの一部は、
  パーミッションが定義されていたとしても使用できない
  - 同じ署名を持つパッケージか、あるいは
    システム領域にインストールされたパッケージに限定される
  - 先の説明と同様にシステム領域に
    インストールする方法もあるが、もっと簡単に…

破れる保護：パーミッション (2)



- root は、すべてのパーミッションを与えられている
  - パッケージのセキュリティチェックは省略
  - この例では、サイレントインストールが可能になってしまう
    - GingerMaster が間接的に使用する

root 化の悪用：対策とその限界 (1)

- 脆弱性を積極的に発見、除去する
  - 当たり前に見えるが、しかし脆弱性が発見されて
    10 ヶ月以上経過してもそれが修正されていない機種が存在する
    (http://www.ipa.go.jp/about/technicalwatch/pdf/110622report.pdf)
- root を制限する：Linux Security Modules (LSM)
  - SHARP 製 Android 端末：Deckard / Miyabi LSM
    - システムパーティションが不正にマウントされることを阻止
    - ptrace (強制的なデバッグ) の禁止など
    - DroidDream や DroidKungFu の感染を阻止する
  - root 権限の悪用を防止する
    - この LSM の保護だけでは侵入の余地があるのだが…

root 化の悪用：対策とその限界 (2)

- root ユーザーの制限だけでは完全な対策にならない
  - パーミッションによる保護は root に対して効果がない
  - Dalvik ベースのアプリはすべて単一プロセスのクローンであるため、既存のセキュア OS のポリシーで保護することが難しい
  - ウイルス対策ソフトの権限は弱い一般権限のまま
- Android 特有の事情を考慮した上で権限の保護を行うことが必要
- 正当なウイルス対策ソフトの権限を、少なくとも root 化したマルウェアと同等まで引き上げられることが望ましい

まとめ

- Android 向けのマルウェアもウイルス対策ソフトも、
  初期と比べると飛躍的に進化している
  - しかし現状ウイルス対策ソフトは低い権限で動作しており、
    限定的な保護しかできていない
- root 化は Android のセキュリティ機構を破り、
  ウイルス対策ソフトが手出しできないマルウェアを可能にしてしまう
  - 仮にウイルス対策ソフトがマルウェアを発見できても、
    それを除去、無害化することができない可能性がある
  - 対処には権限面での見直しや、Android 特有の事情を考慮した
    新しい保護の仕組みが必要とされる

Android は “守れる” か?

総括

Android は守られているか? (1)

- 脆弱性攻撃
  - Android は WebKit など多数のネイティブライブラリに依存しており、ネイティブコード部分は従来と同様に攻撃可能である
  - 本来 Android が基盤とする Linux カーネルの保護機能は、様々な理由によって効果が半減している
    - フレームワークビルド時のミス (DEP)
    - メモリ削減のための機能 (Prelinking)
  - よって脆弱性が存在した場合、それを攻撃するのは現状それほど難しくない
  - ただし、Android 4.0 で状況が変わるかもしれない。

Android は守られているか? (2)

- マルウェア / ウイルス対策ソフト
  - マルウェア (トロイの木馬) はインストールされると、Android 仕組みを用いてスパイウェア、バックドア、あるいはダイアラーなどとして動作する
  - root 化は、ウイルス対策ソフトを役立たずにすることができる

- 現状、Android 端末を十分に守りきることはかなり難しい

これからのためにすべきこと (1)

- 技術的責任 : Android プロジェクト
  - セキュリティ機構を厳密なものに変える
    (root を特別扱いしないようにする)
    - システムコールレベル (LSM)
    - Android フレームワークのセキュア化
  - 正当なウイルス対策ソフトの保護を支援する
    - 例: 特別な署名が成されたアプリに一定の特権を与える
  - メモリ保護機能を強化する
    - Prelinking の見直しやセキュリティ強化
    - …は成されたようだ!

これからのためにすべきこと (2)

- 技術的責任 : 端末メーカー
  - 既存の脆弱性を素早く修正し、アップデートを行う
    (端末ユーザーが攻撃されることをいち早く防ぐ)
  - 独自のカスタマイズを十分に検証する
    - Android のセキュリティ機構が破れないように
      (あるいは正当なプログラムの実行を阻害されないように)

結論

- 現状 Android の保護は十分でなく、また脅威も氾濫しているが、解決が不可能な問題ではない
  - ただし、利用者は当面注意を必要とする
  - ある意味では “割り切ること” も必要かもしれない
- Android プロジェクト、端末メーカー、セキュリティベンダーなどが協力して Android のセキュリティを高めることに注力すべき

ありがとうございました

Fourteenforty Research Institute, Inc.
株式会社 フォティーンフォティ技術研究所
http://www.fourteenforty.jp