脆弱性を含む広告ライブラリの検知機構

A detection mechanism for ad libraries containing vulnerabilities

坂倉友哉,†, 福田浩章 ‡
Tomoya Sakakura†, and Hiroaki Fukuda‡
† 芝浦工業大学　工学部
†Faculty of Engineering, Shibaura Institute Technology.

要旨

Android アプリケーション開発時に，開発者は収入のため広告ライブラリを利用してアプリケーションに広告を組み込む．広告ライブラリの中身を開発者が確認することはほとんどないため，開発者が意図せぬところで広告ライブラリが利用者にとって不利益な動きをすることがある．具体的には, 広告ライブラリとアプリケーションが同一のプロセスで動作することによる権限の共有で生じる個人情報の漏洩や, それを利用した攻撃である．本研究では，脆弱性を生む可能性のある広告ライブラリを，AndroidOS がアプリケーションの実行ファイルのバイナリから検知し，アプリケーションの起動を警告，または中断する機構を提案する．

1. はじめに

多くの Android アプリケーション開発者は収益を得るために広告ライブラリをアプリケーションに組み込む．アプリケーション開発者は，広告ライブラリ提供者と契約することで広告ライブラリに関するソフトウェア開発キットを入手できる. これをアプリケーションに組み込むことで，アプリケーションに広告が現れ，アプリケーション開発者に収益が発生する，広告ライブラリ開発者は，広告ライブラリの挙動に関する公式ドキュメントを開発時に書くが，そのドキュメントが挙動を完全に網羅しているとは限らない [1]．また，アプリケーション開発者も，広告ライブラリの公式ドキュメントを読んで挙動を確認するとは限らない．そして，アプリケーションを利用するユーザは，Google Play ストアでアプリケーションをダウンロードする時，アプリケーションが利用する権限についてなどの詳細ページをみることはほとんどない．つまり，Android アプリケーションに組み込まれた広告ライブラリは，開発者や利用者が想定していない挙動をする可能性がある．

そこで本研究では AndroidOS がアプリケーションに組み込まれている広告ライブラリを自動的に検知し，アプリケーションの起動を警告，または中断する機構を提案する．

2. 広告ライブラリの引き起こす問題

Android アプリケーションは端末のいくつかの機能を利用するためにユーザにシステム権限を要求することがある．例えば，カメラアプリケーションは撮影した写真を保存するために端末のアルバムへのアクセス権限を要求する．システム権限には，カレンダー，カメラ，位置情報，電話，ストレージなどの権限がある．アプリケーションに組み込まれた広告ライブラリは，アプリケーションと同じプロセスで動作するため，アプリケーションに許可した権限を広告ライブラリも使用することができる．広告ライブラリが広告を表示しようとする時，広告ライブラリは広告サーバに広告表示の要求と同時に，デバイスやユーザの情報を送信する．この時，アプリケーションに位置情報の権限を与えた場合，広告ライブラリはユーザの位置情報を広告サーバに送信することができる．また，悪意ある攻撃者が広告サーバに成りすまし，広告データの代わりに任意のコードを送信して実行させることが可能である．これにより，アプリケーションのユーザの端末から任意の番号に電話をかけることや，カレンダーや端末に保存されている連絡先を取得し，書き換えることが可能である [2] .

3. 関連研究

いくつかの広告ライブラリの検知手法が提案されている．[3] は，HTTP 通信と広告描画の周期性から検知する．一般に，広告ライブラリは画面に表示する広告画像を周期的に変更する．その時，広告ライブラリは広告サーバに広告画像を要求し，画像を画面に反映する処理を行う. この時サーバとの HTTP 通信と，画面描画の更新の周期から，アプリケーションに広告ライブラリが組み込まれているか否かを判定する．[4] は，広告ライブラリの広告画像をサーバから取得する挙動に注目し，通信のセッションデータから広告ライブラリを判定する．[5] は，広告ライブラリをリバースエンジニアリングし，得られ

たデータからセマンティック分析と機械学習を用いて広告ライブラリを検出する．ホワイトリストベースの検出が不安定であるとし，その代替手段として提案している [5].

[3][4] は広告要求の動作からの解析であるため，広告サーバにユーザのデータが送信されることになってしまう．[5] は，95 %という高い精度を持ちつつも，ホワイトリストベースでない以上偽検出の可能性が出てきてしまう．また，[3][4][5] では AndroidOS での実装や，広告ライブラリの引き起こす問題を防ぐ手段は考えられていない．よって，実際にユーザが使用することを想定し，Android 端末で広告ライブラリを検知し，その危険性を知らせる機構が必要である．

4. アプローチと実装

本研究では Android 端末の使用を想定するため，ユーザの情報を広告サーバに送信する危険性のない静的解析で検知するシステムが必要である．よって偽検出の起こらないホワイトリストベースでの検知手法を選択する．Android アプリケーションの実行ファイルである dex ファイルを解析することで広告ライブラリを検出し，危険性を通知した上でユーザにアプリケーションの実行を委ねる．また，広告ライブラリの検知はアプリケーションのタップを契機に開始する．これは，アプリケーションが起動してしまっては本研究の意図が失われてしまうこと，およびアプリケーションを起動する方法が少ないのに対し，インストールする方法が多く，実際に 1 つのインストール方法では本機構が機能しないことを確認したためである．

4.1.dex ファイル

Android アプリケーションは，dex ファイルとしてコンパイルされ，Dalvik Virtual Machine で動作する．dex ファイルには，コンパイルされた Java ファイルの名前などが存在する．本研究では問題を起こすことが知られている広告ライブラリの Java ファイル名を dex ファイルから検出し，広告ライブラリを検知する．

4.2.apk ファイル

apk ファイルとはアプリケーションの本体であり，Google プレイストアなどでのアプリケーションインストール時にダウンロードされる．dex ファイルは，アプリケーション作成時に apk ファイルという形式としてマニフェストファイルなどと一緒に圧縮される．apk ファイルは，ダウンロード，インストール時に Android 端末の/data/app/以下に配置される．

4.3. アプリケーションの起動プロセス

アプリケーション起動前に広告ライブラリを解析するには，Android アプリケーションの起動プロセスを変更する必要がある．図 1 にアプリケーションの起動手順を示す．アプリケーションをタップすると，ランチャーが Activity Manager に対してインテントと呼ばれる，アプリケーションの名前や，タップして起動されたかなどが含まれている情報を与える．その後，Activity Manager が Zygote プロセスに起動要求を出し，Zygote が自身を複製してアプリケーション実行プロセスを生成する．生成されたアプリケーション実行プロセスは Activity Manager にインテントを要求し，その情報を利用してアプリケーションが起動する．本研究では，Activity Manager がランチャーからアプリケーションの情報を受け取ってから，Zygote に新規プロセスの生成要求を出すまでの処理に変更を加える．

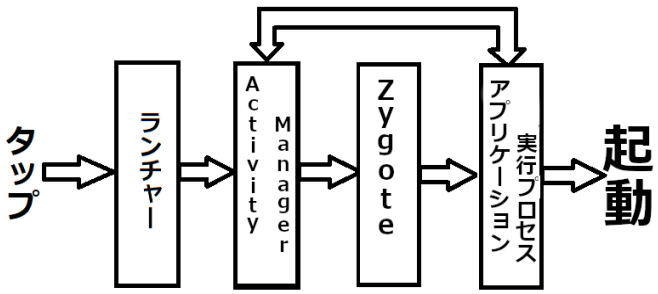

図 1: アプリケーションの起動手順

4.4. 検知手法

図 2，およびソースコード 1，2，3 に変更した処理手順とソースコードの一部を示す．図 2 の 1 はソースコード 1 に対応しており，3 行目の if 文でシステムのデフォルトアプリケーションではないか，タップしてアプリケーションが起動されたかを判定し，true ならばインテントの中身から起動するアプリケーションのパッケージ名を取得し，アプリケーション起動プロセスのプロセス ID と共に新しいスレッドのアプリケーション解析メソッドに与える．図 2 の 2 に対応するソースコード 2 では，アプリケーションのパッケージ名とプロセス ID を受け取った新規スレッドは，まず 2 行目の crawl 関数によってパッケージ名をもとにディレクトリを探索し，apk ファイルを発見する．4 行目の unzip 関数で，発見した apk ファイルを解凍し，5 行目の findDex 関数で出力された classes.dex ファイルを読み込む．広告ライブラリを示す記述を発見した場合は図 2 の 3 に進み，ダイアログを表示してユーザに危険性を示し，アプリケーションの起動を中断するかしないかを問う．中断を選択した場合，17 行目の onClick 関数によりアプリケーションの起動プロセス ID を終了させる．この検知機構を実装し，Nexus5X 実機での動作を確認している．

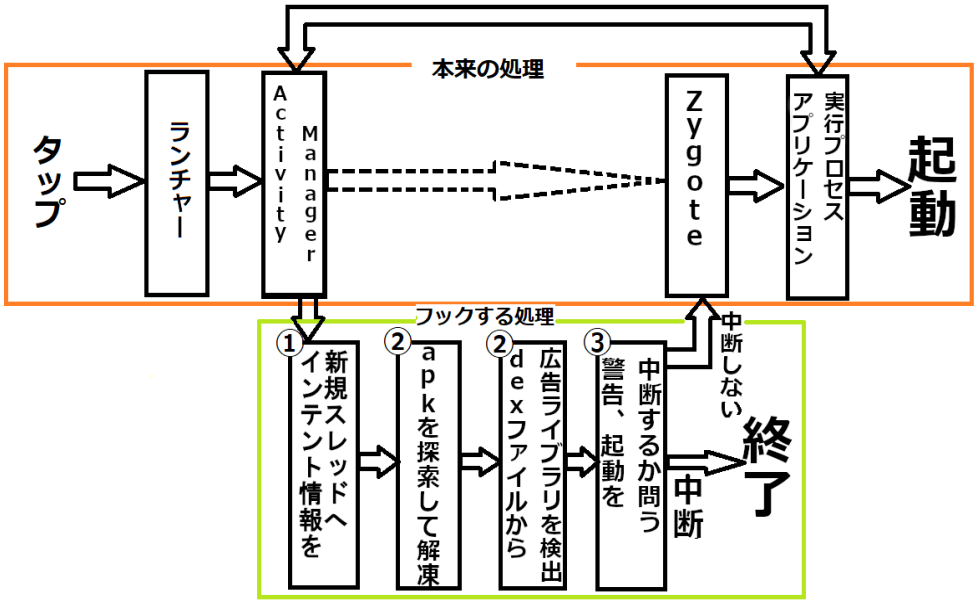

図 2: 改変後の処理手順

```
1    String intentStr = intent.toString();
2    int ppid = Process.myPid();
3    if(intentStr.contains("android.intent.action.MAIN") && !intentStr.
         contains("com.android.") && !intentStr.contains("category.HOME"))
         {
4      // インテントから名前を文字列として取得する処理
5      Log.d(TAG,"logdayonnAMS:apkName:"+apkName);
6     new Thread(new Runnable(){
7        public void run(){
8          CrawlDex cd = new CrawlDex();
9          cd.alerting(apkName,ppid);
10       }
11     }).start();
```

ソースコード 1: 新規スレッドに apk 名とプロセス ID を与える

```
1  public void alerting(String apkName,int ppid){
2    String apkPath = crawl(apkName,rootDir);
3    try{
4      String unzipPath = unzip(apkPath,outputfile1,apkName);
5      findDex(unzipPath,ppid);
6    }catch(IOException e){
7      Log.d(TAG,"ERROR");
8    }
9  }
```

ソースコード 2: apk 解析処理

5. まとめ

Android 端末を利用する際，アプリケーションに組み込まれている広告ライブラリはユーザの情報を漏洩するなどの問題を起こしている．この問題に対して，広告ライブラリを検知する手法は提案されてきたが，実際に Android 端末で動作することを前提にしたユーザを広告ライブラリの起こす問題を回避する提案は存在しない．本研究では実際に Android 端末での動作を目的とし，ユーザの情報が漏洩することのない静的な解析手法を用いた広告ライブラリの検知機構を実装した．しかし，問題を起こす広告ライブラリの名前などは [5] で明らかになっているが，その広告ライブラリを示す dex ファイルの記述との対応付けが時間不足によりできていない．今後は機能の評価のためにこれらの対応付けを行う予定である．

参考文献

[1] Ryan Stevens, Clint Gibler, Jon Crussell, J. Erickson, Hao Chen , Investigating User Privacy in Android Ad Libraries , https://web.cs.ucdavis.edu/ hchen/paper/most2012ad.pdf , 2012

[2] Paul Pearce,Adrienne Porter Felt,Gabriel Nunez,David Wagner , AdDroid: Privilege Separation for Applications and Advertisers in Android , ASIACCS '12 2012 , 2012.

[3] 梶原直也, 川本淳平, 松本晋一, 堀良彰 , 通信と描画挙動の動的解析を用いた Android 広告ライブラリの検知手法 , 2015 年暗号と情報セキュリティシンポジウム , 2015.

[4] Hiroki Kuzuno , Kenichi Magata., Detecting advertisement module network behavior with graph modeling. In Proceedings of the 9th Asia Joint Conference on Information Security(AsiaJCIS2014). IEEE, 2014.

[5] Annamalai Narayanan,Lihui Chen,Chee Keong Chan., AdDetect: Automated detection of Android ad libraries using semantic analysis, 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014.