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

梶原, 直也
九州大学 | 九州先端科学技術研究所

川本, 淳平
九州大学 | 九州先端科学技術研究所

松本, 晋一
九州大学 | 九州先端科学技術研究所

堀, 良彰
佐賀大学 | 九州先端科学技術研究所

他

出版情報：2015年暗号と情報セキュリティシンポジウム. 2015, pp.4A1-1-, 2015-01-20
バージョン：
権利関係：

SCIS 2015 The 32nd Symposium on Cryptography and Information Security
Kokura, Japan, Jan. 20 - 23, 2015
The Institute of Electronics,
Information and Communication Engineers

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
Detection of Android Ad Library Focusing on HTTP Connections and View Object Redraw Behaviors

梶原直也 *† Naoya Kajiwara
川本淳平 *† Junpei Kawamoto
松本晋一 *† Shinichi Matsumoto
堀良彰 ‡† Yoshiaki Hori
櫻井幸一 *† Kouichi Sakurai

あらまし 近年，Android 向け無料アプリケーションの数が急速に増加している．これら無料アプリ開発者の多くは広告ライブラリを用いて開発の報酬を得ている．しかし，広告ライブラリが引き起こす様々な問題点について報告がなされている．そのため，アプリに広告ライブラリが組み込まれているか事前に検知する技術が必要である．関連研究として，アプリのコードにおける特徴を用いた機械学習による広告ライブラリの検知手法が提案されている．しかし，こうした静的解析にはコードの難読化に対応できないといった欠点がある．また，アプリが行なう通信動作に着目した広告の検知手法が提案されているが，通信が暗号化されていた場合この手法は適用できない．本稿では，HTTP 通信と View の描画動作に着目した広告ライブラリの検知手法を提案する．広告ライブラリは広告画像を一定の間隔毎に更新するという特徴に注目した．アプリが行なう HTTP 通信と View の描画動作を記録し，これら 2 つの動作呼び出し時間における周期性を比較することで，アプリに広告ライブラリが組み込まれているか判定する．また，実際にサンプルアプリを用いて提案手法の評価を行った．

キーワード Android，広告ライブラリ，動的解析，フーリエ変換，相関係数

1 はじめに

現在，130 万個以上の Andoird アプリケーション (以下，アプリとする) が Google Play 上で公開されている [1]．これらのアプリの内，約 84%のアプリは無料で公開されている．そのような無料アプリの開発者の多くは，アプリ自体の価格を無料にする代わりに，自身が開発したアプリに広告ライブラリを組み込むことによって，開発の対価を得ている [4]．しかしながら，広告ライブラリが持ついくつかの問題点が明らかになっている．

プライバシ情報の漏洩は，広告ライブラリによって引き起こされる典型的な問題である．これは，広告ライブラリが組み込まれたアプリが動作する際に，ユーザからの同意を得ることなしにユーザの情報を外部へ送信するという問題である．例えば，いくつかの広告ライブラリがユーザの位置情報へアクセスし，外部へ送信しているという報告がある [6]．こういった問題の背景には，アプリ開発者と広告ライブラリ製作者が違うという事実が存在する．広告ライブラリの挙動に関しては，制作元の説明や，ドキュメントを参照することによって，ある程度把握することができる．しかし，アプリ開発者のプライバシ問題への関心の低さ，不十分なドキュメントと言った原因から，アプリ開発者が挙動不明のライブラリを自身のアプリに組み込むという事態が発生する．このように，開発者自身が組み込んだ広告の挙動を正確に把握していないため，予期せぬ情報漏洩が引き起こされる．また，アプリ開発者自身が広告の挙動を把握していない場合，マーケットにおける開発者によるアプリの挙動説明が機能しなくなるという問題にも繋がってしまう．このように，アプリ開発者とユーザの両方にとって挙動不明のアプリが多く流通しているのが現状である．そのため，どのような広告ライブラリが組み込まれているかを事前に検出，分類する技術は重要である．

本稿では，Android 広告ライブラリの検知手法を提案する．我々は，広告ライブラリの基本動作である広告画像の更新動作に着目した．はじめに，広告が組み込まれたいくつかのアプリを動作させ，広告ライブラリが一定

* 九州大学，福岡県福岡市西区元岡 744 番地
† 九州先端科学技術研究所，福岡市早良区百道浜 2-1-22 福岡 SRP センタービル 7 階
‡ 佐賀大学，佐賀県佐賀市本庄町 1 番地

の間隔毎にサーバからの画像取得と画像のスクリーンへの反映動作を行っていることを確認した．AndroidOS を修正し，アプリが行なう HTTP 通信と View オブジェクトの描画動作を記録することによって，広告の取得動作を確認した．通信と描画動作の呼び出し時間データをフーリエ変換し，2 つのデータ間の周期性を比較することで，アプリに広告ライブラリが組み込まれているか判定できることを示す．提案手法を用いた場合，約 95%の検知率が得られた．

2 背景知識

2.1 広告ライブラリ

一般に，広告ライブラリは画面の一部を占有し，そこに広告画像を表示する．ユーザが広告画像をクリックした場合，アプリストアや専用 Web ページへの誘導が行われる．
広告ライブラリはアプリケーション本体のコードと同一パッケージ内で実装される．そのため，Android システム上において，アプリ本体と広告ライブラリは同一プロセス上で動作することになる．このことが，アプリ本体の動作呼び出しと広告ライブラリによる動作呼び出しの区別を困難なものとしている．

2.2 Activity

Activity は，Android アプリの画面に相当するコンポーネントである．Activity とアプリが持つ各画面は一対一に対応する．

2.3 View

View は，Android アプリにおける画面の構成要素に対応するコンポーネントである．テキストやボタン，画像など画面上に配置される全ての要素は View クラス，もしくは View の子クラスによって実装される．これらの View オブジェクトは，そのオブジェクトを配置したいと考えるアプリ画面に対応した Activity オブジェクトにセットされる．この時，Activy オブジェクトに配置される View オブジェクトは木構造の形をとる．図 1 は，Activity において構成される View の木構造の例を表したものである．

3 関連研究

[3] は，Android 向け広告ライブラリの検知手法を提案した論文である．著者らは，本論文の提案手法と同様，広告ライブラリが実行する広告画像の取得動作に着目している．アプリが行なう HTTP 通信をキャプチャし，通信セッションデータからグラフを生成することによって，著者らは広告ライブラリによる広告画像の取得動作を検知している．しかしながら，通信が暗号化されていた場合には適用できないといった課題があった．

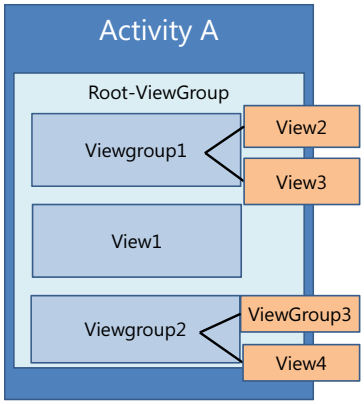

図 1: View によって構成される木構造の例

[5] は，広告ライブラリによって引き起こされる権限共有問題の解決方法を提案した論文である．広告ライブラリには，アプリ本体と権限を共有してしまうという問題点が存在した．アプリ本体のコードと広告ライブラリは同じパッケージにおいて実装され，Android システムにおいて同一プロセス上で実行されるため，このような問題が発生してしまう．この問題を解決するため，アプリ本体と広告ライブラリ間の権限分離手法を著者らは提案している．広告の機能を提供する特別なシステムサービス実行させることによって，このシステムは達成できる．これによって，アプリに広告ライブラリが組み込まれているかの判断が容易になり，権限をより厳格に運用できる．

[7] は，UI 要素に基づいた Android アプリのテスト手法を提案した論文である．著者らは，アプリの画面を構成する要素である Activity と View オブジェクトに着目した．全ての Activity と View オブジェクトを探索することによって，SmartDroid は重要な API 呼び出しにつながる UI に基づくイベントを自動で検知することが出来る．結果として，既存の手法では検知できなかったいくつかの Android マルウェアを SmartDroid が検知できることを著者らは示した．

4 アプローチ

広告ライブラリの検知に関するアプローチについて本節で説明を行う．提案手法では，広告ライブラリの基本動作である広告画像の取得動作に着目した．

4.1 広告ライブラリの挙動に関する仮定

一般に，広告ライブラリは画面上に表示される広告画像を一定の時間間隔で別の画像へと切り替える．本稿で

は，この画像更新動作を以下の 2 つの動作に分別した．

1. HTTP 通信
   広告サーバからの広告画像の取得は HTTP 通信によって実行される

2. View の再描画
   取得した広告画像の画面への再配置に当たる

それから，我々はこれら 2 つの動作呼び出し時刻を記録し，アプリに広告ライブラリが組み込まれているかどうかを自動で判定するシステムを設計した．2.1 節で述べたように，アプリ本体のコードと広告ライブラリのコードは同一プロセス上で実行されるため，これら 2 つの動作を区別することは困難である．しかしながら，我々の仮定では，広告の挙動には何らかの周期性が含まれるの対し，アプリ本体の挙動には周期性が含まれないものとする．これは，一旦広告が画面にセットされると，広告画像の更新が一定の間隔でもって実行されることによる．すなわち，あるアプリに広告ライブラリが組み込まれていた場合，HTTP 通信と View の描画動作呼び出しが一定の周期を持つことになる．以上より，これら 2 つの動作に同一の周期が存在するか確認することによって，広告の有り無しを判定できる．図 2 は，広告ライブラリの動作呼び出しにおける周期性について表す．

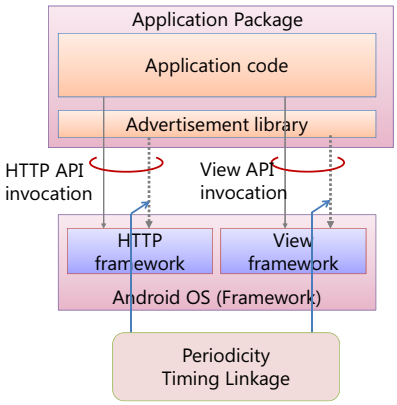

図 2: 広告ライブラリの挙動における周期性

4.2 HTTP 通信と View 描画動作の監視

HTTP 通信と View の描画動作呼び出しを記録するために，AndroidOS の修正を行った．修正した箇所は，アプリ開発者に API を提供する Android フレームワーク層と Java ライブラリ，apache ライブラリである．

4.2.1 HTTP 通信の記録

Android アプリにおける HTTP 通信の実装について，以下の 2 つのクラスを使用することが Android Developers 内で推奨されている [2]．

- org.apache ライブラリにおける HttpClient クラス
- java.net ライブラリにおける HttpURLConnection クラス

しかしながら，アプリ開発者がこれらのクラスを用いずに HTTP 通信を実装する場合にも対応するため，我々は HTTP 通信動作の記録に関するコードを次のメソッドに挿入した．

- java.net ライブラリにおける Socket.connect (SocketAddress endpoint, int timeout) メソッド

また，Android Developer 内で推奨されている上記 2 つのクラスを使用した際にも，Socket.connect メソッドが内部で呼び出さることは確認済みである．

4.2.2 View 描画動作の記録

View の描画動作呼び出しの記録は，android.view.ViewTreeObserver クラスを用いて実装した．2.3 節で述べたように，Android システム上では，アプリの各画面は Activity オブジェクトとそれぞれ対応しており，各 Activity オブジェクトの下に View オブジェクトが木構造の形で配置される．ViewTreeObserver クラスは，この木構造を監視し，新しい View オブジェクトの追加や View の設定変更といったイベントを取得する機能を持つ．我々は，Android フレームワークにおける Activity クラスのソースコードを修正し，以下の操作を実装した．

1. Activity オブジェクトの生成時に，その Activity 内において root となる View オブジェクトを取得する．

2. ViewTreeObserver オブジェクトを root となる View オブジェクトに対してセットする．

3. View の木構造の状態変更時に，ViewTreeObserver.onGlobalLayoutListener メソッドが呼び出される．

4. イベントリスナ内に実装したロギング用のコードが呼び出され，View の描画が実行されたことを示すメッセージを記録する．

4.3 予備実験

第 4.1 節でも述べたように，提案手法においては，アプリが呼び出す HTTP 通信と View の描画動作の周期性を用いて広告の検知を行いたい．本節では，我々が行った予備実験の結果について述べる．予備実験として，以下の 3 つのアプリを動作させ，HTTP 通信と View の描画動作の呼び出し時間にどのような特徴が現れるかを確認した．

- 広告入りアプリ A
  広告以外に HTTP 通信と View の描画動作を呼び出す機能を持っていない
- 広告入りアプリ B
  広告に加えてアプリ本体も HTTP 通信と View の描画動作を呼び出す機能を持つ
- 広告無しアプリ C
  アプリ本体だけが HTTP 通信と View の描画動作を呼び出す

4.3.1 広告入りアプリの動作特徴

広告入りアプリ A 図 3 は，広告入りアプリ A をエミュレータ上で動作させて得られた，HTTP 通信と View の描画動作の呼び出し時間を表したものである．このグラフにおいて横軸は時間を表しており，アプリ A の起動時間から始まり 5 分間を期間としている．HTTP 通信と View の描画動作が呼び出された場合，呼び出しが行われた時間に対応するタイムスロット上に縦棒を記入している．
この図を見ると，HTTP 通信と View の描画動作が約 30 秒の間隔でもってそれぞれ定期的に呼び出されていることが分かる．それぞれの動作呼び出しは，広告画像取得のための HTTP 通信と，取得した画像の画面上への配置動作に相当すると考えられる．

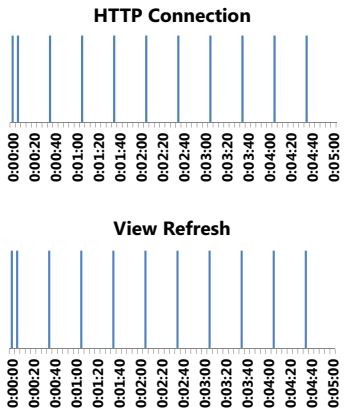

図 3: 広告入りアプリ A を動作させた際の HTTP 通信と View の描画動作の呼び出し時間

広告入りアプリ B 図 4 は，広告入りアプリ B をエミュレータ上で動作させて得られた，HTTP 通信と View の描画動作の呼び出し時間を表したものである．この図を見ると，HTTP 通信に関しては約 30 秒間隔で定期的に呼び出されていることが分かる．一方で，View の描画動作に関しては一見何の周期性もないように見える．これは，HTTP 通信についてはアプリ本体のコードに実装されていないのに対して，View の描画動作はアプリ本体に実装されていないことが原因である．結果的に，View の描画動作呼び出しデータに関しては，広告ライブラリが呼び出したものとアプリ本体が呼び出したものが混在してしまっている．言い換えると，View の描画動作のグラフにおいては，周期性のあるデータ系列と周期性の無いデータ系列が混ざっている．

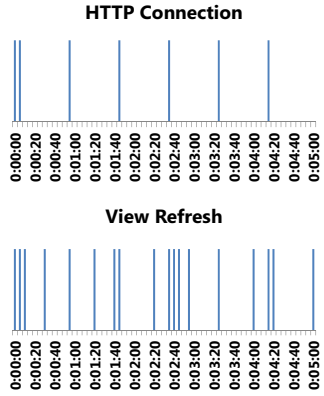

図 4: 広告入りアプリ B を動作させた際の HTTP 通信と View の描画動作の呼び出し時間

4.3.2 広告無しアプリの動作特徴

図 5 は，広告無しアプリ C をエミュレータ上で動作させて得られた，HTTP 通信と View の描画動作の呼び出し時間を表したものである．この図において，HTTP 通信と View の描画動作の呼び出しは何度も行われている．しかしながら，ログデータの全期間を通して，これら 2 つの動作呼び出しには特定の間隔で呼び出されていない．言い換えると，これら 2 つの動作には周期性がないといえる．

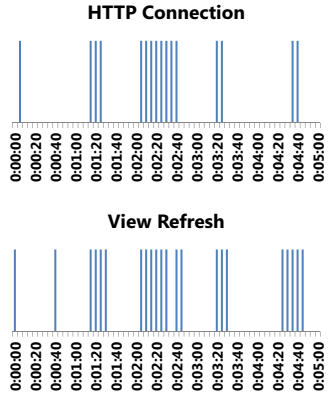

図 5: 広告無しアプリ C を動作させた際の HTTP 通信と View の描画動作の呼び出し時間

4.4 HTTP 通信と View 描画挙動についての考察

本稿において，解析対象となるのはHTTP 通信と View の描画動作の呼び出し時間という 2 つの時系列データである．4.1 節において述べたように，広告ライブラリがアプリに組み込まれていた場合，これら 2 つの挙動に同一の周期が現れると考えられる．しかし，Android システムにおいては，アプリ本体の挙動と広告ライブラリの挙動は区別できないため，周期性のあるデータ系列と周期性の無いデータ系列が混在することとなる．図 4 は，こういったケースの実例である．以上のことから，以下の要件を満たす解析手法が必要である．

1. 2 つの時系列における周期性の比較を行うことが可能
2. 周期性の無いデータが混ざっていたとしても解析を行なうことが可能

5 提案手法

4.1 節において述べたように，広告ライブラリがアプリに組み込まれていた場合，これら 2 つの挙動に同一の周期が現れる．本節では，HTTP 通信と View の描画動作の呼び出し時間という 2 つの時系列データを用いた，広告ライブラリの検知手法について提案する．
4.4 節で述べた要件を満たすような解析手法として，フーリエ解析と相関係数による 2 つの時系列データ間の周期比較を提案する．時系列データをフーリエ変換し，周波数領域でこれらのデータを解析することで，周期をもつ広告ライブラリのデータ系列だけを抽出し，またアプリの動作ラグの影響などを減らすことが可能となる．
図 6 は，提案手法のフローを表している．本節の以降の項では各ステップについて詳細を説明する．

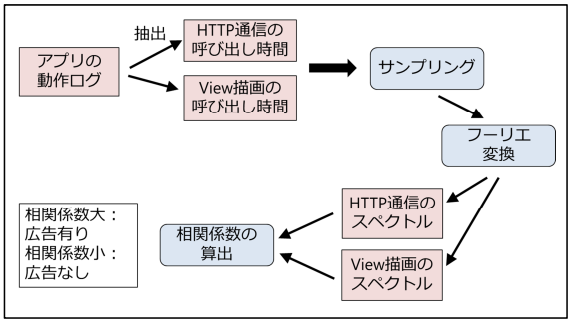

図 6: 提案手法のフロー

5.1 データのサンプリング

初めに，フーリエ変換の前処理としてアプリのログデータのサンプリングを行なう．図 7 は，サンプリングしたデータの例である．データ中の 1 列目は時間を表し，アプリを起動した時間を 0 秒とする．サンプリングデータにおいて，時間はタイムスロットとしてある一定の期間毎に区切られる．図 7 は，タイムスロット幅を 5 秒と設定した場合のデータである．2 列目は，各タイムスロットにおける HTTP 通信呼び出しの有無に対応している．各行のタイムスロットにおいて，アプリが HTTP 通信を行っていた場合”1”を出力し，HTTP 通信を行っていなかった場合”0”を出力している．3 列目は，各タイムスロットにおける View の描画動作呼び出しの有無に対応している．2 列目と同様，各業のタイムスロットにおいて，アプリが View の描画を行っていた場合”1”を出力し，View の描画を行っていなかった場合”0”を出力している．以降の節では，これら 2 つの時系列データをフーリエ変換し，周波数領域での解析を行う．

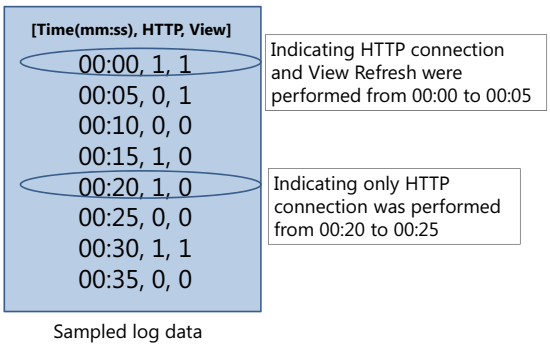

図 7: タイムスロット幅を 5 秒に設定した場合のサンプリングデータの例

5.2 フーリエ変換

ログデータのサンプリングを行った後，サンプリングデータのフーリエ変換を行なう．それゆえ，以降のステップで解析対象となるデータは，横軸を周波数，縦軸を振幅とする周波数スペクトルとなる．図 8 は，図 4 で示したアプリ B のログデータをフーリエ変換して得られた周波数スペクトルを表す．この図において，横軸は周波数のラベルを表す通し番号になっている．

5.3 相関係数を用いた周波数スペクトルの比較

5.2 節で得られた周波数スペクトルを用いた広告の検知を行なう．本稿では，HTTP 通信と View の動作呼び出し時間に関する 2 つの周波数スペクトルの相関係数を用いることで，広告ライブラリの検知が実現できると考

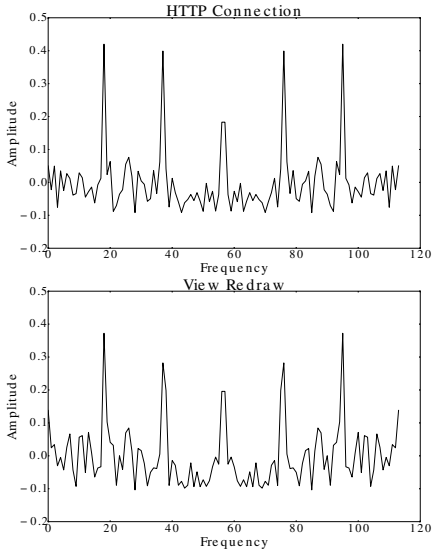

図 8: 広告ありアプリの動作呼び出しデータから生成された周波数スペクトルの例

えた．式 1 は，相関係数の定義を表している．提案手法では，図 8 における 2 つの出力波形を，それぞれ式 1 における x と y に適用した．以前にも述べたように，アプリに広告が組み込まれていた場合，これら 2 つのデータに周期が現れる．フーリエ変換後の周波数スペクトルにおいては，この周期は特定の周波数成分として現れる．図 8 中の波形に現れている 5 つのピーク値がこれに相当する．相関係数を計算する際，各 x と y の値から平均値が引かれている．そのため，5 つのピーク値のみが相関係数の計算結果に大きな影響と与え，他の振幅 0 近傍の値はあまり影響を与えないことになる．したがって，広告入りアプリのログデータを解析した場合，相関係数の値は 1 に近い高い値を取るはずである．
各アプリのログデータに対して相関係数の値を算出した後，ロジスティック回帰を用いた提案手法の検知率に関する評価を行った．

$$\frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}} \tag{1}$$

6 評価実験

6.1 実験手順

本稿では，25 個の広告入りアプリの動作ログデータと，23 個の広告なしアプリのログデータを実験のために準備した．これらのログデータは，HTTP 通信と View の描画動作を記録するように修正した OS を搭載した Android エミュレータ上で，各アプリを動作させることによって作成した．今回の実験では，Google Play マーケットで公開されている無料アプリの内，Android エミュレータ上で正常に動作したアプリだけを使用している．例えば，グラフィックに関するリソースを大量に要求するゲームアプリや，Google Map API を利用するアプリは Android エミュレータ上で動作しないことが確認された．そのため，これらのアプリのログデータは，今回の実験では使用していない．
実験では，提案手法を用いた際の広告入りアプリの検知率と誤検知率について評価を行った．ログデータの少なさから生じる誤差を減らすために，交差検定を用いる．広告入りアプリのログデータと広告なしアプリのログデータを，それぞれ 3 つのグループに分割し，その内 1 つのグループを学習用データ，残りの 2 つのグループをテスト用データとして用いて検知率を算出する．それから，学習用データとテスト用データの組み合わせをローテーションさせることで，3 通りのデータ組み合わせによる評価を行った．
また，実際に解析に用いたデータは，アプリ起動時から最初の 1 分間のデータを削除したものを用いている．これは，広告ライブラリが起動時に初期動作を行い，これがログデータの周期性に影響を与えることを防ぐためである．実際には，最初の 1 分間のデータを削除した後，10 分間分のログデータを解析に用いている．

6.2 解析対象とする広告ライブラリの種類について

本稿において収集したログデータを調査した所，広告ありアプリ 25 個の内，最初に広告画像を表示した後，通信と画面の描画動作を一度も呼び出さないアプリが 4 個存在した．これらのアプリに関しては，本提案手法による解析の対象外としている．これは，解析対象となるデータが存在しないためフーリエ変換が実行できないためである．提案手法では，通信と描画の動作呼び出し時間データに対してフーリエ変換を実行する．そのため，動作呼び出し時間データが存在しない場合，フーリエ変換が実行できず，解析対象とすることができない．こうした画像を更新しない種類の広告ライブラリをどうやって検知するかは今後の課題である．

6.3 実験結果

評価実験において，ログデータのサンプリングを行う際のタイムスロット幅をパラメータとして変更している．今回は，1 秒，2 秒，5 秒，10 秒の 4 通りのパラメータを用いて，それぞれ検知率の評価を行った．また，最終的に実験に使用したアプリの数は，広告ありアプリ 23 個と，広告なしアプリ 23 個である．表 1 は，実験結果を表している．

表 1: Detection Rate and False Positive for Four Parameters

タイムスロット幅	検知率 (%)	誤検知率 (%)
1s	91.67	16.98
2s	94.86	14.62
5s	93.24	18.64
10s	90.26	17.62

6.4 考察

6.4.1 タイムスロット幅による影響

本稿では，ログデータのサンプリング時におけるパラメータとして，4 種類のタイムスロット幅に対して実験を行った．実験結果から，以下のことが考えられる．まず，パラメータを変更したとしても，検知率に大きな影響は与えていないことが分かる．ほとんどの広告は，30 秒から 1 分間程の間隔で広告画像の更新を行っているようである．そのため，タイムスロット幅を数秒単位で変更しても検知率の値に大きな影響を与えなかったものと考えられる．

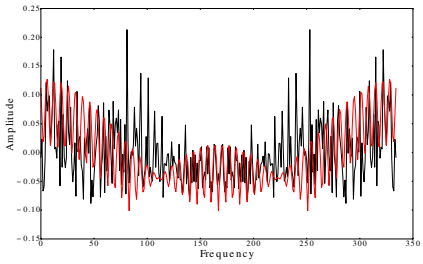

図 9: 誤検知したアプリの周波数スペクトル

6.4.2 誤検知の原因

誤検知の原因について考察を行なう．誤検知してしまったアプリのログデータを見てみると，HTTP 通信と View の描画動作を頻繁に呼び出していたことが確認できた．図 9 は，誤検知したアプリの周波数スペクトルの一例である．このことから，アプリに広告ライブラリが組み込まれていなかったとしても．動作の呼び出し時間に何らかの周期成分が現れてしまった可能性がある．

7 結論

本稿では，HTTP 通信と View の描画動作に着目した広告ライブラリの検知手法を提案した．スマートフォン向け広告ライブラリは，広告画像を定期的に切り替える．このことから，広告ライブラリが組み込まれたアプリを動作させた場合，HTTP 通信と View の描画動作の呼び出し時間は特定の周期を持つことになる．フーリエ変換と相関係数を組み合わせ，この周期を抽出することによって，我々は広告ライブラリの検知手法を実現した．この手法を用いることで，アプリの挙動による広告ライブラリの検知を行なうことが可能である．提案手法では広告の動作に着目したため，広告ライブラリに関するシグネチャやリストを前もって作成する必要が無い．それゆえ，提案手法では未知の広告にも対応することができる．提案手法を用いることで，

謝辞

本研究の一部は，JSPS 科研費 26330169 の助成を受けたものである．本研究を進めるにあたり，有益な助言をいただいた株式会社 KDDI 研究所セキュリティグループ諸氏に感謝する．

参考文献

[1] Android operating system statistics - appbrain. `http://www.appbrain.com/stats/stats-index`.

[2] Connecting to the network — android developers. `http://developer.android.com/training/basics/network-ops/connecting.html`.

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

[4] Ilias Leontiadis, Christos Efstratiou, Marco Picone, and Cecilia Mascolo. Don't kill my ads!: balancing privacy in an ad-supported mobile application market. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 2. ACM, 2012.

[5] Paul Pearce, Adrienne Porter Felt, Gabriel Nunez, and David Wagner. Addroid: Privilege separation for applications and advertisers in android. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 71–72. ACM, 2012.

[6] Ryan Stevens, Clint Gibler, Jon Crussell, Jeremy Erickson, and Hao Chen. Investigating user privacy in android ad libraries. In *Workshop on Mobile Security Technologies (MoST)*, 2012.

[7] Cong Zheng, Shixiong Zhu, Shuaifu Dai, Guofei Gu, Xiaorui Gong, Xinhui Han, and Wei Zou. Smartdroid: an automatic system for revealing ui-based

trigger conditions in android applications. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 93–104. ACM, 2012.