AndroidにおけるWebViewのWebアクセス観測機構を利用した悪性Webサイトの脅威分析と対策の提案

今村 祐太[1]　折戸 凜太郎[2]　Kritsana Chaikaew[2,3]　Celia Manardo[1,4]
Pattara Leelaprute[3]　佐藤 将也[1]　山内 利宏[1,a)]

概要：多くのAndroidアプリケーション（以降，Androidアプリ）がWebViewを利用し，Androidアプリ内でWebコンテンツを表示している．WebViewとは，Webブラウザに切り替えることなく，Androidアプリ内でのWebコンテンツの表示を可能にするコンポーネントである．しかし，WebViewは攻撃に利用される可能性がある．また，我々が調査した限りでは，WebViewを悪用した攻撃を防ぐためにアクセス制御を提案している先行研究はあるものの，WebViewの通信内容に着目し，それを分析した先行研究はない．そこで，本稿では，この問題に対処するために，文献[1], [2]で提案したAndroidにおけるWebViewのWebアクセス観測機構を利用し，WebViewを介した通信の脅威を分析した結果を報告する．本稿では，特に3種類のWebアクセスの脅威を明らかにし，これらに対する対策を提案する．また，WebViewにおける対策技術の実現可能性を確認した実験の結果を報告する．

キーワード：Android，WebView，ネットワーク監視，JavaScript

Yuta Imamura[1]　Rintaro Orito[2]　Kritsana Chaikaew[2,3]　Celia Manardo[1,4]
Pattara Leelaprute[3]　Masaya Sato[1]　Toshihiro Yamauchi[1,a)]

1. はじめに

多くのAndroidアプリケーション（以降，Androidアプリ）において，WebViewが利用されている．WebViewとは，Androidアプリ内にWebコンテンツを表示する際に利用するコンポーネントである．WebViewの利用により，利用者はWebブラウザを利用することなく，Androidアプリ内でWebサイトを閲覧できる．また，Androidアプリ開発者はWebViewを利用することで，WebView内にロードされたJavaScriptからAndroid端末の資源を操作できる高機能なアプリを開発できる．上記の理由から，WebViewは多くのAndroidアプリで利用されている．例えば，FacebookやTwitterの公式アプリでURLをタップすると，WebViewを介してAndroidアプリ内にWebサイトが表示される．先行研究では，Googleが管理するアプリストアのAndroidアプリについて，2011年時点で約86%[3]，2014年6月時点で85%[4]のAndroidアプリがWebViewを利用していることを報告している．

一方，WebViewは，Webブラウザ同様にJavaScriptを実行できることから，Webアプリケーションの脆弱性を悪用する攻撃手法であるクロスサイトスクリプティング攻撃は，WebViewにおいても有効である[5], [6], [7]．また，WebViewのaddJavascriptInterface APIを悪用した攻撃，JavaScriptによる攻撃，およびAdSDKを悪用した攻撃が報告されている[8], [9]．さらに，悪性Webサイトへアクセスした国内のモバイル端末利用者数が増加傾向にある[10]．文献[10]では，攻撃者がソーシャルメディア上の投稿やメッセージ，および不正広告により利用者を悪性Webサイトに誘導し，不審なアプリをインストールさせる攻撃が報告されている．上述の通り，FacebookやTwitterの公式アプリにおいてもWebViewが利用されているため，利

[1] 岡山大学 大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University
[2] 岡山大学 工学部
Faculty of Engineering, Okayama University
[3] カセサート大学 工学部
Faculty of Engineering, Kasetsart University
[4] Engineering School of Grenoble INP (National Polytechnic Institute) group
[a)] yamauchi@cs.okayama-u.ac.jp

用者は悪性 Web サイトによる被害を受ける可能性がある．以上より，WebView を利用する Android アプリにおける WebView を介した通信の脅威を分析し，対策を施す必要がある．しかし，我々が調査した限りでは，WebView を悪用した攻撃を防ぐためにアクセス制御を提案している先行研究はあるものの，WebView を対象とした観測機構や WebView の通信内容に着目し，それを分析した先行研究はない．

この問題に対処するために，我々は，Android における WebView の Web アクセス観測機構（以降，提案機構）を提案した [1], [2]．提案機構は，WebView を利用する Android アプリ毎に，WebView を介した通信を収集し，保存することができる．また，tcpdump や Wireshark は SSL/TLS によって暗号化された通信内容を平文として取得できない．一方，提案機構は SSL/TLS によって暗号化された通信内容を平文として収集できる．さらに，提案機構は，WebView を介して Web アクセスした Android アプリを特定できる情報として，Android アプリのパッケージ名を収集できる．以上より，提案機構は，WebView を介した通信の脅威分析に有用であるといえる．

本稿では，提案機構を利用し，WebView を介した通信の脅威を分析した結果を報告する．特に，3 種類の Web アクセス（偽警告画面，コインマイニング，およびフィッシング）の脅威を明らかにし，これらに対する対策を報告する．また，WebView における悪性 Web サイトへのアクセスを検知・防止する機構の実現可能性を確認した実験の結果を報告する．なお，WebView を介した通信の脅威を分析するために，Facebook と Twitter の公式アプリを利用した．

本研究の主な貢献は，以下に示す通りである．

- 提案機構を利用して WebView を介した通信の脅威を分析した．また，特に脅威となる 3 種類の Web アクセス（偽警告画面，コインマイニング，およびフィッシング）に着目して，これらの Web アクセスを分析し，以下の内容を明らかにした．
  (1) WebView を介した Web アクセスにおける偽警告画面を表示する仕組み
  (2) WebView で Web アクセスした際の “Coinhive” の危険性
  (3) WebView を標的としたフィッシング攻撃の危険性と特徴
- 3 種類の Web アクセスの脅威分析の結果から，これらの Web サイトに対する WebView を介した通信の対策を提案する．また，WebView 内で特定 Web サイトへのアクセスを検知・防止する実験を実施し，WebView を介した悪性通信の検知・防止機構の実現可能性を示した．

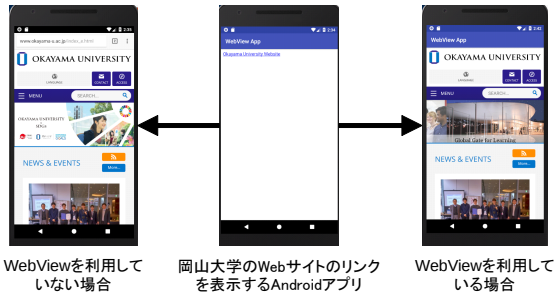

図 1　WebView の利用例

2. 背景

2.1 WebView

WebView とは，Web ブラウザに切り替えることなく，Android アプリ内での Web コンテンツの表示を可能にするコンポーネントである．**図 1** に WebView の利用例を示し，以下で詳細を述べる．岡山大学の Web サイトのリンクを表示する Android アプリが WebView を利用していない場合，Web ブラウザが起動される．一方，WebView を利用している場合，Web ブラウザは起動されず，Android アプリ内に岡山大学の Web サイトが表示される．

近年，有名な Android アプリに限らず，多くの Android アプリが WebView を利用している．有名な Android アプリの例として，Twitter や Facebook といった SNS アプリ，Gmail のようなメールアプリ，および Uber のような地図を利用するアプリがある．以上から WebView は，Android アプリにとって欠かせないコンポーネントであるといえる．しかし，我々が調べた限りでは，WebView のセキュリティに着目した研究はあまりない．

また，WebView には様々な API が存在する．Android アプリ開発者は，これらの API を利用することで，より利便性の高い Android アプリを開発することができる．Android アプリと Web ページを連携させる際に頻繁に利用される API を以下に示す．

(1) loadUrl API : 指定した URL の Web コンテンツを WebView 内にロードし，表示する．
(2) setJavaScriptEnabled API : WebView 内にロードした JavaScript の実行を有効化する．
(3) addJavascriptInterface API : WebView 内にロードした JavaScript から Android 端末内で定義されているメソッドを実行することができる．

2.2 WebView の Web アクセス観測機構

我々は，WebView を介した通信を収集し，その通信内容を分析するために，提案機構を実現した [1], [2]．提案機構を利用することで，WebView を介した Web アクセスを観測し，Web アクセスの分析に必要な情報を収集することができる．提案機構が取得できる情報を**表 1** に示す．提案

表 1　提案機構が取得可能な情報

	取得する情報
通信内容を把握できる情報	HTTP リクエスト HTTP レスポンス
通信先を特定できる情報	通信先の URL 通信先の IP アドレス 通信先のポート番号
どの Android アプリによる通信かを特定できる情報	Android アプリのパッケージ名

表 2　分析の評価環境

Android エミュレータ	Android 6.0
WebView	60.0.3094.2

機構は，表 1 に示す情報を各 Android アプリに割り当てたデータ領域に保存する．これにより，Android アプリ毎に収集した通信ログを基に WebView を介した通信を分析することができる．また，提案機構は，WebView が HTTP リクエストを暗号化する前に HTTP リクエストを取得し，WebView が HTTP レスポンスを復号した後に HTTP レスポンスを取得する．これにより，提案機構は，SSL/TLS によって暗号化された通信内容を平文として収集できる．

3. 提案機構を利用した悪性 Web サイトの分析

3.1 分析の目的と考え方

2.1 節で述べたように，WebView は多くの Android アプリにおいて利用されている．また，WebView は Web ブラウザ同様に JavaScript を実行できる．さらに，WebView には様々な API が提供されており，これらの API を利用することで，Android アプリと Web コンテンツの連携を容易に実現できる．

しかし，WebView は攻撃に利用される可能性がある．WebView は，Web ブラウザ（例えば，Chrome や Firefox）とは異なり，実装は，Android アプリ開発者に委ねられる．このため，Android アプリ開発者がセキュリティを考慮せず Android アプリを開発した場合，攻撃者によって Android アプリに存在する脆弱性が悪用される可能性がある．また，我々が調査した限りでは，WebView を悪用した攻撃を防ぐためにアクセス制御を提案している先行研究があるものの，WebView の通信内容に着目し，それを分析した先行研究はない．

そこで，本研究では，Android の Web アクセスにおける脅威を分析し，特に WebView を介した Web アクセスの脅威を明らかにすることを目的とする．WebView を介した通信の脅威を分析するために，提案機構を利用して WebView を介した通信を収集し，分析した．なお，WebView を利用しており，さらに幅広く利用されているアプリとして，Twitter アプリと Facebook アプリから WebView を介した通信を収集し，収集した通信内容を分析した．評価環境は，表 2 に示す通りである．

3.2 Android の Web アクセスにおける脅威

Android の WebView 経由の Web アクセスの脅威について，調査対象を絞る事前調査として，Android における脅威についてまとめてある Web ページなど [10] の調査，および最近の脅威動向の調査を行った．また，公開されている悪性 Web サイトのブラックリストに，WebView 経由で実際にアクセスし，WebView において脅威となる悪性 Web サイトを調査した．これらの結果から，WebView において特に脅威となる以下の 3 種類の脅威に着目し，提案機構を利用して収集した通信ログを詳細に分析した．

(1) 偽警告画面
画面の任意の場所をタップすることで，偽警告画面を表示する Web サイトへと利用者を誘導する Web サイトが存在する．これは，Web ブラウザだけでなく WebView を利用するアプリにおいても発生する．本稿では，偽警告画面を表示する Web サイトへのリダイレクト方法と偽警告画面を表示する方法について述べる．

(2) コインマイニング
仮想通貨を採掘（コインマイニング）するサービスとして，“Coinhive” が存在する．Coinhive は，Web サイトを訪問した利用者の端末を利用し仮想通貨を採掘する．本稿では，WebView で Web アクセスした際の Coinhive の危険性について述べる．

(3) フィッシング
WebView を利用する Android アプリには，基本的に URL バーが存在しないため，Web ページの URL を確認することが難しい [11]．このため，WebView におけるフィッシング攻撃において，利用者が正規の Web サイトと偽の Web サイトを区別するのは困難である．本稿では，WebView におけるフィッシング攻撃の危険性とフィッシング攻撃の特徴について述べる．

以降では，3.3 節で各脅威の分析方法について述べ，3.4 節以降で分析した結果を述べる．

3.3 分析方法

3.3.1 偽警告画面

偽警告画面を表示する Web サイトが存在する．文献 [10] では，偽警告画面を表示する Web サイトの目的は，広告収入の獲得や不審な Android アプリのインストールが目的であると述べられている．また，ワンクリック詐欺ソフトや情報窃取型不正アプリがインストールされ，金銭が要求される被害も報告されている．そこで，偽警告画面を表示する Web サイトとその Web サイトへと利用者を誘導する Web サイトの通信ログの収集・分析から，WebView を利用した Web アクセスにおける偽警告画面の表示の仕組み

を明らかにし，WebView によるブラウジング時にその脅威を防止することを目的とする．また，通信ログの収集には，提案機構と tcpdump を利用した．以下に，分析手順を示す．

(1) 提案機構と tcpdump による通信ログの収集を有効にする．
(2) Twitter アプリもしくは Facebook アプリを起動し，偽警告画面が表示される URL をタップし，WebView の機能によりアクセスする．
(3) 収集した通信ログを用いて，リンククリック時の挙動を把握するために次の方法で分析する．
  (a) 手動で通信ログを分析，特に JavaScript ファイルを抽出し，攻撃内容を分析する．
  (b) Wireshark を利用して PCAP ファイルを分析する．

3.3.2 コインマイニング

提案機構を利用し，Facebook アプリにおける WebView を介した通信を収集している際に，WebView においても Coinhive が実行されることを確認した．本分析では，WebView を介したアクセス時に，Coinhive を利用した仮想通貨を採掘する手法の分析を目的とする．以下に，分析手順を示す．

(1) Facebook アプリを起動し，Coinhive の設置された Web サイトに WebView の機能によりアクセスする．
(2) 提案機構を利用し，Facebook アプリにおける WebView を介した通信を収集する．
(3) 手動で通信ログを分析する．

3.3.3 フィッシング

フィッシング攻撃は，Web ブラウザだけでなく WebView を利用する Android アプリにおいても起こりうる．本分析では，WebView を利用する Android アプリにおけるフィッシング攻撃の分析を目的とする．以下に，分析手順を示す．

(1) フィッシングサイトを見つけるために，Twitter アプリ上でキーワードとして “click here” と “free phone” を検索する．
(2) Twitter アプリを起動し，見つかったフィッシングサイトに WebView の機能によりアクセスする．
(3) 提案機構を利用して，フィッシングサイトの通信を収集する．
(4) 手動で通信ログを分析する．

3.4 偽警告画面

3.4.1 概要

ある Web サイト（以降，遷移元サイト）は，閲覧中に**図 2** のような警告画面を表示する Web サイトへと利用者を誘導する．図 2 の警告画面の目的は，広告収入の獲得や不審な Android アプリのインストールである．また，警告画面の中には，表示される際にバイブレーションを発生させるものや音声を流すものもある．さらに，一度警告画面が

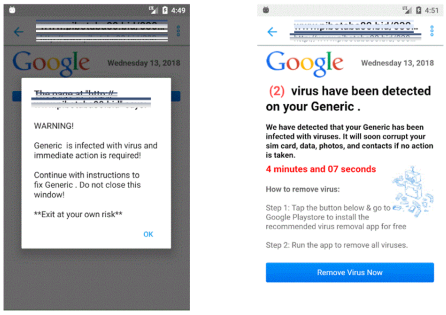

図 2 偽警告画面（言語を英語に設定した場合）

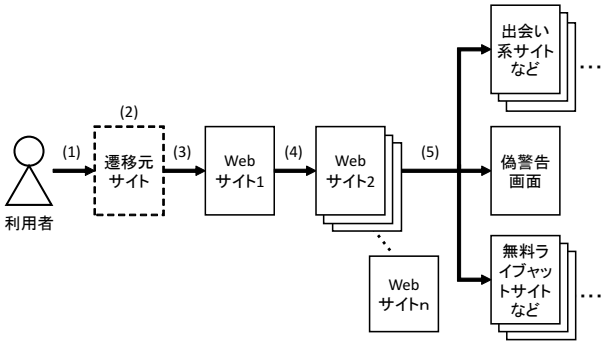

図 3 偽警告画面を表示する際の Web サイトの遷移の流れ

表示されると，利用者は他の Web ページに遷移できない．戻るボタンを押しても反応せず，画面のどの領域をタップしても，Android アプリのインストールサイトに移動する．これにより，利用者はあたかも Android 端末がマルウェアに感染したような錯覚に陥り，Android アプリをインストールする可能性がある．また，偽警告画面を表示する JavaScript を埋め込まれた Web ページの内容は表示されることがなく，偽警告画面表示前の Web サイトに戻ることができない．

偽警告画面の表示による Android アプリインストール画面への誘導手法について，以下の内容を調査し，脅威を明らかにした．

(1) 偽警告画面へリダイレクト手段
(2) 偽警告画面へのリダイレクト条件
(3) リダイレクトされる Web サイト
(4) 偽警告画面の表示方法
(5) 広告ブロック機能の回避手法

3.4.2 偽警告画面へのリダイレクト手段

図 3 に偽警告画面を表示する際の Web サイトの遷移の流れを示し，以下でその詳細を述べる．

(1) 利用者が遷移元サイトを訪問
  利用者が遷移元サイトを訪問する．
(2) 画面の任意の場所をタップ
  利用者が遷移元サイトの任意の場所をタップする．
(3) Web サイト 1 へのリダイレクト
  遷移元サイトは，利用者のタップを検知し，利用者を Web サイト 1 にリダイレクトする．なお，利用者は，

```
if(!this.utils.cookieEnabled()){return false}
this.settings=this.utils.merge(this.defaults,window.__htapop?window.__htapop:{});

cookieExpires:1

if(typeof(c[f])==="object"&&c[f].length>0){
for(var e=0;e0||(document.cookie="test").indexOf.call(document.cookie,"test")>1)){},
event:{ add:function(target,name,handler)
```

図 4　Cookie の文字を含む JavaScript コード

遷移元サイトのどの部分をタップしても，Web サイト 1 へリダイレクトされる．

(4) Web サイト 2 へのリダイレクト
Web サイト 1 にリダイレクトした後，利用者を再度 Web サイト 2 にリダイレクトする．このリダイレクトには，JavaScript “window.location.replace” を利用する．この JavaScript は，現在表示している Web ページの履歴を残さずに指定した URL に利用者をリダイレクトさせることができる．これにより，利用者は Android 端末の「戻る」ボタンを押下しても 1 つ前のページに戻れなくなる．

(5) 偽警告画面を表示する Web サイトへのリダイレクト
Web サイト 2 にリダイレクトした後，偽警告画面を表示する Web サイトにリダイレクトする．このリダイレクトには，JavaScript “window.location.href” を利用する．なお，偽警告画面を表示する際の JavaScript の分析結果は以降の項で述べる．

3.4.3　偽警告画面へのリダイレクト条件

遷移元サイトから偽警告画面が表示されるまでの通信を収集している際，偽警告画面が表示される場合と表示されない場合があった．この原因を調べるために，偽警告画面が表示される場合の通信ログと表示されない場合の通信ログを比較し，**図 4** に示す JavaScript コードを発見した．この JavaScript コードから偽警告画面へのリダイレクト条件として，Cookie が利用されていると推察できる．

また，我々は偽警告画面を表示する Web ページへの遷移に Cookie が関連していることを確認するために，以下の実験を行った．次に示す条件において，Google Chrome，Twitter アプリ，および自作アプリで遷移元サイトにアクセスし，Cookie の保持や取得の有無による挙動の違いを調査した．なお，(C) で利用する自作アプリは，WebView を利用しており，かつ Cookie を使用しないアプリである．

条件

(i) 遷移元サイトの Cookie を保持していない場合
(ii) 遷移元サイトの Cookie を保持している場合
(iii) 遷移元サイトの Cookie を消去した場合

上記の実験から，以下の結果を得ることができた．

(i) 遷移元サイトの Cookie を保持しておらず，Cookie を取得する場合
遷移元サイトの任意の場所のタップにより，意図しない遷移が発生する．

```
&geo='JP'
&geocode='Japan'
&isp='Research Organization of Information and Systems'
&states='Okayama'
&city='Okayama'
&brand='Generic'
&browser='Chrome Mobile+'
&os='Android+6.0'
```

図 5　偽警告画面の表示に利用される情報

(ii) 遷移元サイトの Cookie をすでに保持している場合
遷移元サイトの任意の場所のタップによる意図しない遷移は発生しない．

(iii) 遷移元サイトの Cookie を取得できない場合
遷移元サイトの任意の場所のタップによる意図しない遷移は発生しない．

以上の結果から，Cookie を保持しておらず，遷移元サイトから Cookie を取得する場合において意図しない遷移が起きることを明らかにした．

3.4.4　リダイレクトされる Web サイト

偽警告画面を表示する Web サイトにリダイレクトするまでに，複数回リダイレクトが行われた．また，Web サイト 1 において実行される JavaScript では，特定の URL と無作為に生成された 36 進数の文字列を組み合わせた URL を 10 個生成していた．さらに，Web サイト 1 からリダイレクトされる Web サイトは，乱数を生成し，それを利用して 10 個の URL から遷移サイトが選出される．このため，アクセスする度に高い確率で異なる Web サイトにリダイレクトされる．図 3 に示すように，Web サイト 2 は，出会い系サイトや無料ライブチャットサイトなどの様々な Web サイトへリダイレクトしている．また，遷移元サイトからのリダイレクト先や遷移元サイトからのリダイレクト数は毎回同じであるとは限らない．実際に，偽警告画面を表示する Web サイトに遷移しない場合も確認している．

3.4.5　偽警告画面の表示方法

提案機構によって収集した通信ログの分析により，利用者情報（例えば，OS 情報，ブラウザのバージョン，Java のバージョン，および Flash のバージョン）を取得する JavaScript コードを発見した．また，偽警告画面を表示する Web サイトの JavaScript ファイルを見つけた．この JavaScript ファイルは，**図 5** に示す利用者情報（例えば，国名，都道府県，市町村，および OS 情報）を取得し，それを基に偽警告画面を生成する．さらに，この JavaScript ファイルには，カウントダウンタイマを利用するコードや利用者の端末のバイブレーションを動作させるコードも含まれていた．これらの情報と 3.4.3 項で述べた方法を利用して，利用者に合わせた偽警告画面を表示する．これにより，利用者に表示された警告画面が正規の警告画面であると信じ込ませることができる．また，表示される警告画面には Android アプリをダウンロードさせるボタンがある．

```
<script src="https://coinhive.com/lib/coinhive.min.js"></script><script>
<!--//--><![CDATA[// ><!--
var miner = new CoinHive.Anonymous("XoWXAWvizTNnyia78qTIFfATRgcbJfGx",{throttle:
0.1});miner.start();
//--><!]]>
```

図 6 Coinhive の JavaScript

表示された警告画面を正規の警告と信じた利用者は，このボタンをタップし，不審な Android アプリをダウンロードする可能性がある．

また，言語情報を利用して，表示する警告画面の言語を切り替えていることを確認した．実験では，日本語，英語，および中国語で偽警告画面が表示されることを確認した．

3.4.6 広告ブロック機能の回避手法

Web ページに表示される広告の表示をブロックし，Web ページのコンテンツのみを表示するための機能として，広告ブロック機能が存在する．しかし，偽警告画面を表示する Web サイトの中には，WebSocket を利用しているものもある．このため，広告ブロック機能による警告画面の表示を防ぐ機能を回避することができる．

3.5 コインマイニング

3.5.1 脅威

Coinhive は，Web ページ訪問者に広告を表示する代わりに，短い JavaScript コードを実行し，Web サイトの訪問者の端末を利用して，仮想通貨 “Monero” を採掘し，その一部をサイト運営者の収入とすることができる．しかし，攻撃者は，Web サイトを改ざんし，Coinhive を悪用して収入を得ることができる．我々が Coinhive の利用状況を調査したところ，2018 年 3 月時点で約 32,000 の Web サイトにて Coinhive が利用されているという報告があった [12]．

3.5.2 Coinhive を利用した Web サイトの攻撃方法

図 6 に Coinhive の JavaScript を示し，以下で述べる．

(1) JavaScript ファイルのダウンロードとインストール
“script src =” により，Coinhive の Web サイトから JavaScript ファイルをダウンロードし，それを Web サイトに設置させる．

(2) アカウントの作成
Coinhive のサービスを利用するためには，アカウントを作成する必要がある．Web サイトを訪問した利用者のアカウント作成後，変数 “miner” に作成したアカウント情報を代入する．

(3) 採掘の開始
“miner.start” により Monero の採掘を開始する．

上記の方法で Coinhive を Web サイトに設置することができ，攻撃者は秘密裏に Monero を採掘できる．

3.6 フィッシング

3.6.1 脅威

フィッシング攻撃は，Web ブラウザだけでなく WebView を利用する Android アプリにおいても起こりうる．また，WebView を利用する Android アプリには，基本的に URL バーが存在しない．このため，利用者が偽の Web サイトと正規の Web サイトを区別するのは困難である．以上より，WebView を利用する Android アプリにおけるフィッシング攻撃は，Web ブラウザにおけるフィッシング攻撃よりも強力である．また，フィッシング攻撃により，利用者は，偽の Web サイトを正規の Web サイトと思い込み，入力フォームに個人情報を入力してしまう可能性がある．

3.6.2 分析結果

我々は，Twitter アプリのリンクから発見したフィッシングサイトの通信を分析した．分析の結果，これらの Web サイトの多くが，“X-Robots-Tag” ヘッダで “noindex” ディレクティブを利用していることが分かった．このタグは，検索エンジンから Web サイトを隠すために利用される．これにより，Twitter などの SNS アプリのリンクからのみ，その Web サイトにアクセスすることができる．Twitter などの SNS アプリも WebView を利用しており，利用者が Twitter などの SNS アプリのリンクからフィッシングサイトにアクセスした場合，URL バーがないため，利用者が偽の Web サイトと正規の Web サイトを区別するのは困難である．このため，利用者はアクセスした Web サイトを信用し，個人情報を入力してしまう可能性がある．

4. 対策

4.1 偽警告画面

Google Chrome において，Google Safe Browsing API と Yahoo!スマホセキュリティ [13] のセキュリティ機能により，偽警告画面を表示する Web サイトへの遷移を防ぐことができる場合がある．これは，これらのセキュリティ機能が URL をブラックリストと照合することよって悪性な URL への Web アクセスを検知するためである．しかし，これらのセキュリティ機能は，WebView の Web アクセスを対象としていないため，WebView 経由の悪性 Web サイトへのアクセスは検知できない．また，最新の Android 8.0 の WebView において，Google Safe Browsing API がデフォルトで有効になっており，唯一の防止手段となっている．しかし，Google Safe Browsing API の検知率は，今回実験した範囲では高くなかった．

一方，提案機構は WebView 内で URL やメタデータなどを収集している．このため，提案機構に URL だけでなく，Web ページのコンテンツを取得する通信の IP アドレスやメタデータなどの情報を利用するセキュリティ機構を追加することで，悪性 Web サイトへのアクセスを防ぐことができる可能性がある．また，偽警告画面を表示する Web サイトへのリダイレクトと偽警告画面の表示は，JavaScript が原因である．これらの JavaScript をさらに分析し，その特徴を得ることで，WebView 内で悪性な JavaScript を検

知できる可能性がある．

4.2 コインマイニング

Monero を採掘する JavaScript コードが難読化されていない場合，“var miner = CoinHive.Anonymous(”*SITE-KEY*”)” のパターンマッチングにより秘密裏に Monero を採掘する Web サイトへのアクセスを検知・防止できる．これは，WebView に実現した提案機構内で検知機構を実現できる可能性がある．一方，JavaScript が難読化されている場合は，Coinhive のサーバとのコネクションを確立する際に，接続先の IP アドレスを確認することで，Coinhive の Web サイトへのアクセスブロックできる．攻撃者は，Coinhive の JavaScript コードをダウンロードするために，Coinhive の Web サイトに必ずアクセスする必要がある．このため，WebView 内で接続先の IP アドレスを利用してアクセスをブロックすることは有効であるといえる．

4.3 フィッシング

WebView を利用する Android アプリには，基本的に URL バーがないため，WebView におけるフィッシング攻撃において，利用者が正規の Web サイトと偽の Web サイトを区別するのは困難である．WebView を利用する Android アプリにおけるフィッシング攻撃を防ぐためには，Google Safe Browsing などのブラックリスト方式が有効である．しかし，現時点では，Android の最新版しか対応していないことと，検知率が低い問題がある．また，Android 8.0 の WebView において，Google Safe Browsing の利用はデフォルト有効であるものの，Android アプリ開発者が無効にすることが可能であるため，不正アプリの場合，この機能が無効にされている可能性がある．

Android における既存の対策では URL のブラックリストを用いるのに対して，提案機構では HTTP のリクエストとレスポンスを取得しているため，“X-Robots-Tag” が HTTP リクエストに含まれているか否かも判定に利用することで，今回発見したフィッシングサイトへのアクセスをうまく防ぐことができる可能性がある．

5. WebView 内での特定 Web サイトへのアクセスの検知・防止実験

我々は，WebView が特定 Web サイトに HTTP リクエストを送信する前に，提案機構によって取得した情報をブラックリストと照合する機構を提案機構に追加した．提案機構は，HTTP リクエストを送信する前に，表 1 に示す情報のうち，HTTP リクエスト，通信先の URL，通信先の IP アドレス，および通信先のポート番号を取得する．このため，WebView が特定 Web サイトにアクセスする前に，URL もしくは IP アドレスをブラックリストと照合することで，その通信を検知し，それを防止することができる．

本実験は，WebView を介した悪性通信の検知・防止機構の実現可能性を示すことを目的とし，ある広告配信サーバへのアクセスを IP アドレスのブラックリスト方式により検知し，その通信を防止できるか否か検証した．実験の結果，Android アプリに広告が表示されず，収集した通信ログに広告配信サーバからの HTTP レスポンスが含まれていないことを確認した．また，Android アプリがクラッシュすることなく，正常に動作することを確認した．以上より，WebView 内で特定 Web サイトへのアクセスを検知し，その通信を防止することができるといえる．また，本実験結果から，提案機構に悪性 Web サイトへのアクセスを検知し，それを防止する機構を追加することで，WebView を介した悪性通信を検知・防止できる可能性がある．

6. 関連研究

WebView の addJavascriptInterface API を利用することで，WebView 内にロードされた JavaScript はネイティブ言語のメソッドにアクセスすることができる．この仕組みを悪用し，WebView を利用する Android アプリを標的とした攻撃がいくつか報告されている．これらの攻撃への対処や WebView のセキュリティの向上を目的に，WebView を利用する Android アプリを対象としたアクセス制御の研究として，文献 [14], [15], [16], [17] がある．

文献 [14] では，ハイブリッドアプリケーションの JavaScript コードによるデバイス資源へのアクセスを制御するために，ハイブリッドアプリケーションが読み込む URL 毎に開発者がアクセスパーミッションを設定できるアクセス制御方式を提案している．文献 [15] では，addJavascriptInterface API による Java オブジェクトの登録可否を制御することで，JavaScript から Android 端末内の危険な API へのアクセスを制御する方式を提案している．文献 [16] では，WebView を利用した Android アプリにおいて，JavaScript と Java 間の共通のインタフェースを用いて，デバイスの資源を JavaScript 側から操作できるインタフェースの悪用を防止するために，外部の生成元のコンテンツからアプリやデバイス資源へのアクセスを制御する方式を提案している．文献 [17] では，悪意のある攻撃者がリパッケージと呼ばれる手法を用いて，プラグインを悪用する JavaScript コードを Cordova アプリに挿入し，デバイスの資源の奪取や改ざんを行う攻撃を防止するために，プラグインによるデバイスの資源へのアクセスを動的に制御する方式を提案している．

また，WebView 内にロードされた悪性な Web コンテンツによる攻撃が報告されている [11], [18]．文献 [11] では，シンボリック実行と静的解析を利用し，ハイブリッドアプリケーションにおけるイベントハンドラを自動的に検査する EOEDroid を提案している．文献 [18] では，悪性 Web コンテンツによる各 Android アプリの WebView インスタ

ンスを悪用した攻撃を報告し，その攻撃に対する対策として OS レベルでの緩和策を提案している．

上述の文献は，WebView を悪用する攻撃を対象としたアクセス制御，その攻撃の緩和策，およびソースコードレベルでの Android アプリの分析を提案しているものの，WebView の通信内容に着目していない．このため，本稿で述べた 3 種類の Web アクセス（偽警告画面，コインマイニング，およびフィッシング）の脅威を緩和することはできない．これらの Web アクセスの脅威を緩和するためには，WebView を介した通信の脅威を分析し，その分析結果を基に対策を施す必要がある．提案機構は WebView を介した通信を収集することができるため，提案機構は WebView を介した通信の脅威分析に有用である．また，WebView を介した通信の脅威分析の結果から，悪性 Web サイトにおける通信の特徴を得ることで，WebView 内で悪性通信を検知・防止できる可能性がある．

7. おわりに

Android の Web アクセスにおける脅威について調査し，WebView を介した通信の脅威を述べた．また，WebView を介した通信の脅威を明らかにするために，提案機構を利用して WebView を介した通信の脅威分析を実施し，3 種類の Web アクセス（偽警告画面，コインマイニング，およびフィッシング）の脅威を報告した．

3 種類の Web アクセスの分析結果から，これらの Web アクセスに対する対策を提案した．また，WebView に実現した提案機構内で特定の Web サイトへのアクセスを検知・防止する実験を実施し，IP アドレスをブラックリストと照合することで，広告配信サーバへの Web アクセスを検知・防止できることを示した．さらに，検知・防止実験の結果から，WebView を介した悪性通信の検知・防止機構の実現可能性を示した．

残された課題として，WebView を介した悪性通信を検知・防止するための提案機構のさらなる拡張がある．

謝辞 本研究成果の一部は，国立研究開発法人情報通信研究機構 (NICT) の委託研究「Web 媒介型攻撃対策技術の実用化に向けた研究開発」により得られたものです．

参考文献

[1] 今村祐太，上川先之，石原靖弘，佐藤将也，山内利宏：Android における WebView の Web アクセス観測機構，コンピュータセキュリティシンポジウム 2017 論文集，vol.2017，no.2，pp.545–552 (2017).

[2] Imamura, Y., Uekawa, H., Ishihara, Y., Sato, M., Yamauchi, T.: Web Access Monitoring Mechanism for Android WebView, *Proceedings of the Australasian Information Security Conference (AISC 2018)*, pp.1–8 (2018).

[3] Luo, T., Hao, H., Du, W., Wang, Y., Yin, H.: Attacks on WebView in the Android System, *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC 2011)*, pp. 343–352, (2011).

[4] Mutchler, P., Doupe, A., Mitchell, J., Kruegel, C., Vigna, G.: A Large-Scale Study of Mobile Web App Security, *Proceedings of the Mobile Security Technologies Workshop (MoST)*, (2015).

[5] Bhavani, A.B.: Cross-site Scripting Attacks on Android WebView, *Proceeding of the International Journal of Computer Science and Network (IJCSN 2013)*, Vol 2, Issue 2, pp.1–5, (2013).

[6] Luo, T., Du, W., Wang, Y.: ATTACKS AND COUNTERMEASURES FOR WEBVIEW ON MOBILE SYSTEMS, PhD thesis, Syracuse University, (2014).

[7] Bao, W., Zong, M., Wang, D.: Cross-site Scripting Attacks on Android Hybrid Applications, *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy (ICCSP '17)*, pp.56–61, (2017).

[8] Neugschwandtner, M., Lindorfer, M., Platzer, C.: A View To A Kill: WebView Exploitation, *Proceeding of the 6th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 2013)*.

[9] Son, S., Kim, D., Shmatikov, V.: What Mobile Ads Know About Mobile Users, *Proceedings of NDSS 2016*, pp.1–15, (2016).

[10] 岡本 勝之：最新モバイル脅威事情：モバイルこそ「Web 経由」に注意，トレンドマイクロ セキュリティブログ，入手先〈`https://blog.trendmicro.co.jp/archives/13524`〉（参照 2018–08–11）．

[11] Yang, G., Huang, J., Gu, G.: Automated Generation of Event-Oriented Exploits in Android Hybrid Apps, *Proceedings of NDSS 2018*, pp.1–15, (2018).

[12] Brian Krebs: Who and What Is Coinhive?, Krebs on Security, available from 〈`https://krebsonsecurity.com/2018/03/who-and-what-is-coinhive/`〉(accessed 2018–08-20).

[13] スマホの安全対策はこれで OK ！無料で始めるセキュリティアプリ，Yahoo!スマホセキュリティ，入手先〈`https://spsecurity.yahoo.co.jp/promo/index.html#aku-text`〉（参照 2018–08–20）．

[14] Jin, X., Wang, L., Luo, T., Du, W.: Fine-Grained Access Control for HTML5-Based Mobile Applications in Android, *Proceedings of he 16th Information Security Conference(ISC)*, (2013).

[15] Jing, Y., Yamauchi, T.: Access Control to Prevent Malicious Javascript Code Exploiting Vulnerabilities of Webview in Android OS, *IEICE Transactions on Information and Systems*, vol. E98-D, no.4, pp.807–811, (2015).

[16] Tuncay, G. S., Demetriou, S., Gunter, C. A.: Draco: A System for Uniform and Fine-grained Access Control for Web Code on Android, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*, pp. 104–115, (2016).

[17] Kudo, N., Yamauchi, T., Austin, T. H.: Access Control for Plugins in Cordova based Hybrid Applications, *Proceedings of the 31st IEEE International Conference on Advanced Information Networking and Applications (AINA 2017)*, pp. 1063–1069, (2017).

[18] Li, T., Wang, X., Zha, M., et al.: Unleashing the Walking Dead: Understanding Cross-App Remote Infections on Mobile WebViews, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017)*, pp.829–844, (2017).