Computer Security Symposium 2017
23 - 25 October 2017

AndroidにおけるWebViewのWebアクセス観測機構

今村 祐太[1]　上川 先之[1]　石原 靖弘[2]　佐藤 将也[1]　山内 利宏[1,a)]

概要：Android で Web コンテンツを表示する方法として，Web ブラウザ以外に WebView を利用する方法が存在する．WebView とは，Android アプリケーションでの Web コンテンツの表示を可能にするコンポーネントであり，多くの Android アプリケーションで利用されている．一方で，我々が調査した範囲では，WebView に着目した Web アクセス観測方法は存在しない．そこで，WebView のみの Web アクセスを観測する手段として，WebView の Web アクセス観測機構を提案する．本稿では，その設計と実現方式について述べる．また，実現した観測機構の性能を評価し，結果について述べる．

キーワード：Android，WebView，Web アクセス観測

Yuta Imamura[1]　Hiroyuki Uekawa[1]　Yasuhiro Ishihara[2]　Masaya Sato[1]
Toshihiro Yamauchi[1,a)]

1. はじめに

Google が開発しているスマートフォン向けのオペレーティングシステムである Android オペレーティングシステム（以降，Android）を搭載したスマートフォンが普及している．2016 年 11 月に公表された調査結果では，Android の市場占有率は約 88%まで拡大したと報告されている [1]．

Android で Web コンテンツを表示する方法として，Web ブラウザを利用する方法以外に，Android アプリケーション（以降，Android アプリ）に Web コンテンツを表示させる方法が存在する．多くの Android アプリでは，これを WebView と呼ばれるコンポーネントにより実現している．

文献 [2] では，2011 年時点での Google が管理するアプリストアの Android アプリを調査した結果，約 86%の Android アプリが WebView を利用していることが判明したと報告されている．また，文献 [3] では，2014 年 6 月時点での Google Play の Android アプリを調査した結果，85%の Android アプリが WebView を利用していると報告されている．さらに，我々は，上記の調査と比べると小規模ではあるが，2017 年 6 月時点で Google Play で配信されている 32 個のアプリについて，Android アプリのソースコード中の WebView 関連の記述を検索し，Android アプリでの WebView の利用率を調査したところ，約 97%の Android アプリで WebView が利用されていることがわかった．このように，WebView は多くの Android アプリで利用されている．

一方，WebView を利用した Web コンテンツの表示は Web ブラウザへ切り替わらないため，利用者は，Web アクセスが行われていることを認識していない可能性が高い．また，悪意のある Android アプリで WebView が利用されている場合，秘密裏に不正な Web アクセスが行われ，Android 端末内の情報の奪取や改ざんをされる可能性がある．このため，WebView を介した Web アクセスを観測し，分析する必要がある．さらに，WebView を介した Web アクセスにより，外部サイトからダウンロードした JavaScript が実行可能であるため，悪意のある JavaScript 実行についても分析する必要がある．

しかし，我々の調査した限りでは，WebView を悪用した攻撃が報告されている [2][4][5][6][7][8][9] 一方で，WebView を介した Web アクセスの通信に着目した報告はない．また，Android 上の Web アクセスのうち，WebView を介した Web アクセスを特定する既存研究はなく，WebView を介した Web アクセスを観測するための機構もない．Android 上の Web アクセスを観測する方法として，HTTP プロキシやパケットキャプチャツールを利用する方法があるが，

[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
[a)] yamauchi@cs.okayama-u.ac.jp

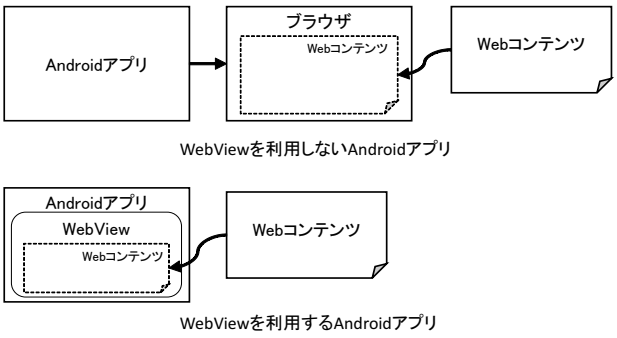

図 1 Web コンテンツの表示方法

Android 上の Web アクセスのうち，WebView を介した Web アクセスを特定することはできない．

そこで，本稿では，現時点で WebView を介した Web アクセスのみを観測する方法がないため，Android における WebView の Web アクセス観測機構を提案し，その実現方式を述べる．提案機構は，Chromium WebView の C++層を改変し，既存の WebView を置き換えることで実現する．これにより，Android 上の Web アクセスのうち，WebView を介した Web アクセスのみを観測可能にする．また，提案機構の実現方式は，Android フレームワークやカーネルの改変を必要とせず，WebView の入れ替えのみで導入できる利点がある．なお，提案機構を適用した WebView を Android 端末に導入する際は，Android 端末を root 化する必要がある．また，本稿では，提案機構の性能を評価した結果について述べる．

2. WebView

2.1 概要

WebView とは，Web ブラウザに切り替えることなく，Android アプリ上での 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，Android で Web コンテンツを表示する 2 種類の方法を示す．図 1 に示すように，WebView を利用しない Android アプリで，Web コンテンツを表示させようとした場合，デフォルトのブラウザアプリに切り替わり，そこで Web コンテンツが表示される．一方，WebView を利用する Android アプリでは，デフォルトのブラウザアプリへの切り替えは行われず，Android アプリ上に Web コンテンツを表示する．

WebView は，Android のバージョン毎に使用しているブラウザエンジンが異なる．Android 4.1〜Android 4.3 (JellyBean) までの WebView は，WebKit[10] を利用している．また，Android 4.4 (KitKat) 以降の WebView は，Chromium[11] を利用しており，Chromium WebView と呼ばれる．

Android 4.4 (KitKat) までの WebView は，Android フレームワーク内に実現されていた．一方，Android 5.0 (Lollipop) 以降の WebView は，Android フレームワークから分離され，システムアプリケーションとして実現されている．これにより，Android をアップデートすることなく，WebView を Google Play からアップデートできるようになった．

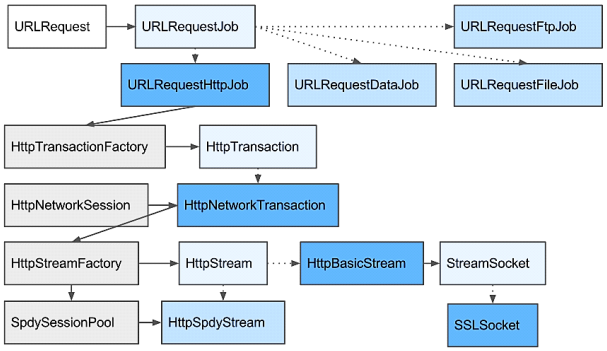

図 2 Chromium をベースとする Web ブラウザのネットワークスタック

2.2 構成

図 2 に Chromium をベースとする Web ブラウザのネットワークスタックを示す．Chromium プロジェクトは，多くのプラットフォーム用の Web ブラウザを提供している．Chromium プロジェクトにより開発されている Web ブラウザは，フロントエンドや機能の有無などの違いはあるが，通信処理部分などのプラットフォーム間でほとんど違いのない部分は，実装が共有されており，図 2 に示すクラス図で構成されている．Chromium をベースとする Web ブラウザによる Web アクセスは，すべて図 2 の URLRequest クラスから始まる．

また，Chromium WebView は，Chromium プロジェクトにより開発されているため，Chromium WebView も同様に URLRequest クラスによって Web アクセスが行われる．Chromium WebView は，Java と C++を用いて開発されており，Java 層と C++層で構成される．ここで，WebView における C++層が Chromium をベースとする Web ブラウザのネットワークスタックの実装に相当する．

2.3 Web アクセスの処理流れ

図 3 に WebView を介した Web アクセスの処理流れを示し，以下でその詳細を述べる．

(1) Web アクセスを行うメソッドの呼び出し
　WebView を利用する Android アプリが，Web アクセスを行うメソッドを呼び出す．ここで，Web アクセスを行う WebView のメソッドとして，loadUrl() メソッド，loadData() メソッド，loadDataWithBaseURL() メソッド，および postURL() メソッドがある [6]．

(2) JNI を利用した WebView における C++層のメソッドの呼び出し
　(1) で呼び出されたメソッドが，JNI を利用し，We-

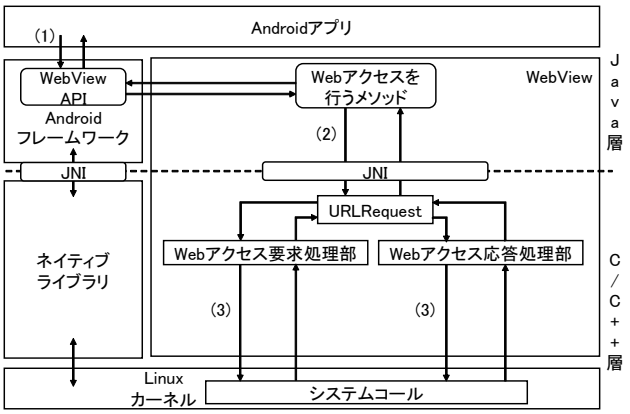

図 3　WebView を介した Web アクセスの処理流れ

bView における C++層のメソッドを呼び出す．ここでは，URLRequest クラスのメソッドを呼び出す．(1) で呼び出されるメソッドは，指定した URL の Web コンテンツの表示のみを行っており，Web アクセスの処理は，WebView における C++層で行われる．

(3) システムコールの発行

Web アクセスを行うために，WebView における C++層のメソッドが，システムコールを発行する．

2.4　問題点

1 章で述べたように，WebView は多くの Android アプリで利用されている．一方で，WebView の利用により，不正な Web アクセスが行われる可能性がある．WebView を利用した Web アクセスは，Web ブラウザへの切り替えが行われず，利用者は，Web アクセスが行われていることを認識していない可能性が高い．このため，不正な Web アクセスが行われた場合においても，その Web アクセスに気付かない可能性がある．

また，悪意のある Android アプリで WebView が利用されている場合，秘密裏に不正な Web アクセスが行われ，Android 端末内の情報の奪取や改ざんをされる可能性がある．さらに，WebView を悪用した攻撃が報告されている．文献 [2][4][5][6] では，addJavaScriptInterface API を利用し，Android 端末内の情報を奪取する攻撃について報告している．文献 [5][7][8] では，WebView を利用する Android アプリを標的としたクロスサイトスクリプティング攻撃について報告している．ただし，文献 [8] は，Android のハイブリッドアプリケーションを標的としている．また，文献 [9] では，AdSDK を利用する Android アプリを対象に，利用者の機密情報を奪取する攻撃について報告している．

一方で，我々が調査した限りでは，WebView を介した Web アクセスのみを観測し，その分析を行ったという報告はなされていない．WebView を介した Web アクセスを分析するためには，WebView を介した Web アクセスを観測し，その通信ログを収集する必要がある．

WebView を介した Web アクセスを観測する方法として，HTTP プロキシやパケットキャプチャツールを利用する方法がある．しかし，この方法は，Android におけるすべての Web アクセスを観測するため，WebView を介した Web アクセスを区別して観測することはできない．そこで，WebView を介した Web アクセスに着目し，その Web アクセスを観測する機構が必要である．

3.　WebView の Web アクセス観測機構

3.1　目的と考え方

WebView を介した Web アクセスを観測する手段として，Android における WebView の Web アクセス観測機構を提案する．提案機構は，Android における Web アクセスのうち，WebView の Web アクセスに着目し，その Web アクセスを観測することを目的とする．

WebView を介した Web アクセスを観測するために，WebView を介した Web アクセスの処理の途中に Web アクセスを観測する機能を追加し，提案機構を実現する．Web アクセスを観測する機能の追加箇所として，以下の 3 つが考えられる．

（追加箇所 1）Android フレームワーク
（追加箇所 2）WebView
（追加箇所 3）Linux カーネル

より多くの WebView を介した Web アクセスを観測するためには，多くの Android 端末に提案機構を導入する必要がある．このため，導入の容易さを考慮する必要がある．上記の追加箇所のうち，Android フレームワークと Linux カーネルに Web アクセスを観測する機能を追加する場合，Android 端末毎に Android フレームワークや Linux カーネルを改変する必要がある．一方，WebView に Web アクセスを観測する機能を追加する場合，WebView の入れ替えのみで提案機構を導入することができる．以上より，WebView に Web アクセスを観測する機能を追加し，WebView を介した Web アクセスを観測する．

3.2　全体像

図 4 に提案機構の全体像を示す．2.3 節で述べたように，WebView を介した Web アクセスは，WebView における C++層の Web アクセス要求処理部と Web アクセス応答処理部によって行われる．そこで，Web アクセス要求処理部と応答処理部に Web アクセスを観測する機能を追加し，提案機構を実現する．これにより，WebView を介した Web アクセスの処理流れを変更することなく，WebView を介した Web アクセスを観測することができる．

3.3　課題

提案機構の課題を以下に示す．また，各課題の対処を 3.4 節で述べる．

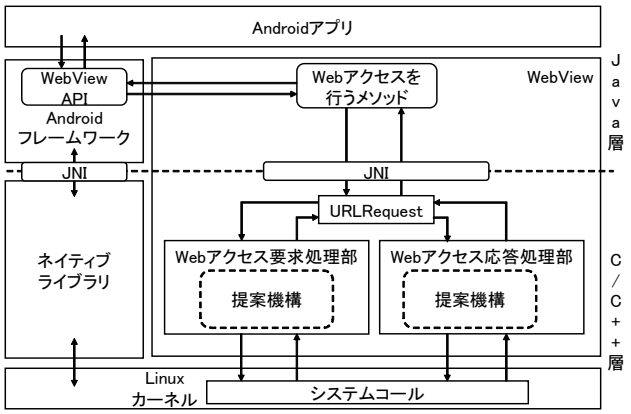

図 4　提案機構の全体像

（課題 1）取得する情報の検討

本研究の目的は，提案機構により収集した WebView の通信ログをもとに，WebView を介した Web アクセスの安全性を検証することを目的とする．また，その通信ログをもとに，悪意のある Android アプリを発見することを目的する．そこで，WebView を利用する Android アプリを分析するために取得する必要がある情報について検討する．

（課題 2）取得する情報の保存形式と保存場所の検討

本研究の目的を達成するために，分析することを考慮し，（課題 1）で検討した情報の保存形式と保存場所について検討する．

3.4　課題への対処

3.4.1　取得する情報の検討

WebView を利用する Android アプリを分析するために以下の情報を取得する．

(i) HTTP リクエストと HTTP レスポンス

Web ブラウザと Web サーバとの間でのデータの転送は，HTTP を利用する．WebView においても，Web アクセスを行う場合，HTTP の形式に従い，データを作成する．このため，HTTP の形式に従い作成されたデータを取得することで Web アクセスの通信内容を把握することができる．以上のことから，WebView を介した Web アクセスの通信内容を把握するために，HTTP リクエストと HTTP レスポンスの情報を取得する．

(ii) タイムスタンプ

WebView を介した Web アクセスは，非同期処理で行われる．このため，HTTP リクエストと HTTP レスポンスのみでは，それらの対応関係を把握することができない．一方，WebView と Web サーバとのコネクション間でのデータ転送は，同期処理で行われる．WebView を介した Web アクセスは，コネクション毎にインスタンスが生成される．そこで，タイムスタンプを取得し，それを各コネクションの識別子として用いる．これにより，HTTP リクエストと HTTP レスポンスの対応関係を把握できる．

(iii) Android アプリのパッケージ名

WebView を利用する Android アプリを分析する場合，どの Android アプリによる Web アクセスかを把握する必要がある．そこで，Android アプリを識別するための情報として，Android アプリのパッケージ名を取得する．

(iv) Web コンテンツの URL

HTTP リクエストに含まれるスキーム名，ホスト名，およびパス名から，アクセス先の Web コンテンツの URL の情報を取得することができる．一方で，WebView を利用する Android アプリを分析することを考慮すると，アクセス先の Web コンテンツの URL を取得する方が良いと考える．そこで，WebView を介した Web アクセスのリクエスト情報として，Web コンテンツの URL を取得する．

(v) 通信先の IP アドレス

WebView を介した Web アクセスのリクエスト情報として，通信先の IP アドレスを取得する．
また，Web コンテンツを悪用する攻撃では，攻撃サイトの IP アドレスやドメイン名を短期間のうちに遷移させることで，ブラックリストによる対策を困難にする手法を利用するものがある [12]．このため，(i) で取得した情報のみを基に攻撃サイトの分析を行った場合，攻撃サイトを特定することは困難である．これは，一定期間後，攻撃に用いられる攻撃サイトのドメインの DNS 登録情報が変更されている可能性が高いためである．攻撃サイトを特定するためには，DNS 登録情報が変更される前の IP アドレスを取得する必要がある．以上のことから，リクエスト情報として，通信先の IP アドレスを取得することで，DNS 登録情報が変更された場合においても，攻撃サイトを特定できる可能性を高くできる．

(vi) Web サーバのポート番号

ネットワーク層における TCP や UDP は，ホスト間通信のエンドポイントを指定するための識別子として，ポート番号を用いている．HTTP を利用した Web アクセスでは，通常は 80 番ポートを利用する．一方，HTTPS を利用した Web アクセスでは，443 番ポートを利用する．このように，プロトコル毎に使用するポート番号が異なる．このため，Web サーバのポート番号を取得することで，どのプロトコルを利用した Web アクセスかを推測できる．

(vii) ソケット接続の際のコネクションエラー

Web コンテンツを悪用する攻撃では，攻撃サイトの IP アドレスやドメイン名を短期間のうちに遷移させる

表 1　提案機構の処理と処理のタイミング

提案機構の追加箇所	処理	取得する情報	タイミング
Web アクセスの要求処理	（処理 1）	ソケット接続の際のコネクションエラー	connect() システムコール処理完了直後
	（処理 2）	タイムスタンプ Android アプリのパッケージ名 HTTP リクエストヘッダ Web コンテンツの URL 通信先の IP アドレス	HTTP リクエストヘッダ生成直後
	（処理 3）	HTTP リクエストボディ	HTTP リクエストボディの送信前
Web アクセスの応答処理	（処理 4）	HTTP レスポンスヘッダ	HTTP レスポンスヘッダ受信完了後
	（処理 5）	HTTP レスポンスボディ	HTTP レスポンスボディ受信完了後

攻撃が存在するため，攻撃に用いられる攻撃サイトが存在する期間は短いという特徴がある．このため，攻撃サイトへ Web アクセスする場合，コネクションエラーが生じる可能性がある．そこで，ソケット接続の際のコネクションエラーの情報も取得する．これにより，正規の Web サイトから攻撃サイトへ改ざんされた Web コンテンツを特定できる可能性を高くできる．

3.4.2　取得した情報の保存形式と保存場所の検討

WebView を介した Web アクセスでは，Web アクセスの要求処理と応答処理が非同期で処理を行う．また，Web アクセスの際に，コネクション毎にインスタンスが生成され，HTTP リクエストと HTTP レスポンスの対応関係を把握することができない．WebView を利用する Android アプリを分析するためには，HTTP リクエストと HTTP レスポンスの対応関係を把握する必要がある．そこで，3.4.1 項で述べた情報をコネクション毎に保存する．取得する情報をコネクション毎に保存する方法として，3.4.1 項で述べた情報のうち，タイムスタンプを利用する方法があり，コネクション毎に生成されたインスタンスで HTTP リクエスト作成時のタイムスタンプを取得し，それをファイル名とすることで，取得した情報をコネクション毎に保存できる．また，分析の容易さと保存する際の軽量さをを考慮した独自の形式で取得した情報を保存し，WebView を利用する Android アプリの分析時に解析用計算機上で取得した情報を JSON 形式に変換する．さらに，3.4.1 項で述べた情報を各 Android アプリ毎に割り当てられたデータ領域に保存する．これにより，WebView を利用する Android アプリ毎に WebView を介した Web アクセスの通信ログを収集することができる．

3.5　提案機構の処理流れ

図 5 に提案機構を適用した WebView の Web アクセスの処理流れを示す．図 5 のうち，提案機構の処理流れの詳細を以下に示す．また，**表 1** に提案機構の処理と処理のタイミングを示す．

（処理 1）connect() システムコールによるコネクションの確立が失敗した場合に，そのエラー情報をソケット接続の際のコネクションエラーとして取得し，Android 端末の内部ストレージに保存する．

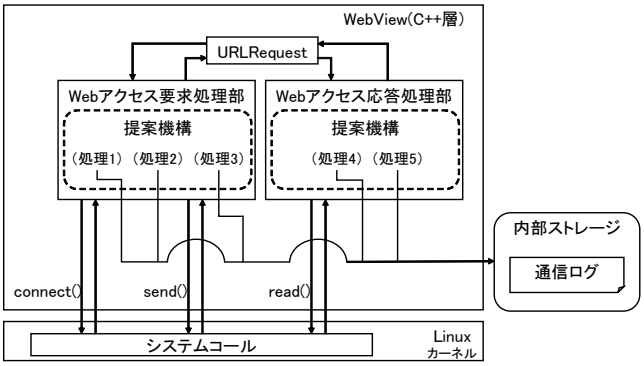

図 5　提案機構を提供した WebView の Web アクセスの処理流れ

（処理 2）HTTP リクエストヘッダ生成直後に，その際のタイムスタンプ，Android アプリのパッケージ名，HTTP リクエストヘッダ，Web コンテンツの URL，および通信先の IP アドレスを取得し，Android 端末の内部ストレージに保存する．

（処理 3）POST メソッドを利用する場合に，送信前に HTTP リクエストボディを取得し，Android 端末の内部ストレージに保存する．

（処理 4）read() システムコールによる HTTP レスポンスヘッダの受信が完了した後，HTTP レスポンスヘッダを取得し，Android 端末の内部ストレージに保存する．

（処理 5）read() システムコールによる HTTP レスポンスボディの受信が完了した後，HTTP レスポンスボディを取得し，Android 端末の内部ストレージに保存する．

3.6　期待される効果

提案機構の導入により，以下の効果が期待できる．

（効果 1）WebView を介した Web アクセスを観測可能

WebView における C++層に Web アクセスを観測する機能を追加し，提案機構を実現する．これにより，Android における Web アクセスのうち，WebView を介した Web アクセスに着目し，その Web アクセスを観測することができる．

（効果 2）WebView の通信ログに着目し，WebView を利用する Android アプリを分析可能

提案機構は，3.4.1 項で述べた情報を WebView を利用する Android アプリ毎に Android 端末上に保存する．また，分析の容易さと保存する際の軽量さを考慮した独自の形式で取得した情報を保存する．これにより，WebView を利用する Android アプリ毎に，WebView の通信ログに着目し，Aroid アプリを分析することができる．さらに，提案機構を適用した WebView をサンドボックスとして使用することで，マルウェアや悪意のある Android アプリの挙動を解析することができる．

（効果 3）HTTPS による暗号化通信の通信内容を観測可能

HTTPS は，アプリケーション層の HTTP を保護するプロトコルであり，Web ブラウザや WebView と Web サーバとの間の通信を暗号化し，通信内容の盗聴や改ざんを防いでいる．提案機構は，HTTP リクエストと HTTP レスポンスを暗号化されていない状態で取得することができる．このため，通信内容が暗号化される前に HTTP リクエストと HTTP レスポンスを取得することができる．

4. 実現方式と評価

4.1 実現方式

3 章で述べた WebView の Web アクセス観測機構を Chromium WebView 60.0.3094.2 に実現した．提案機構は，WebView における C++層のクラスのうち，以下の 2 つのクラスを改変し，実現する．

(1) HttpStreamParser クラス
(2) SocketPosix クラス

(1)HttpStreamParser クラスは，HTTP/1.0，もしくは HTTP/1.1 の形式のデータを生成し，Web アクセスの要求処理と Web アクセスの応答処理を行う．なお，HttpStreamParser クラスでは，HTTP/2.0 を利用する Web アクセスを観測することができないため，提案機構による HTTP/2.0 を利用する Web アクセスの観測は，今後の課題とする．

提案機構は，HTTP/1.0，もしくは HTTP/1.1 の形式の HTTP リクエストが生成された直後に，HTTP リクエスト作成時のタイムスタンプ，Android アプリのパッケージ名，HTTP リクエスト，Web コンテンツの URL，通信先の IP アドレス，および Web サーバのポート番号を取得し，Android 端末の内部ストレージに保存する．（図 5 の（処理 2））．これらの情報のうち，Web コンテンツの URL，通信先の IP アドレス，および Web サーバのポート番号は，他クラスのインスタンスから取得する．また，POST メソッドを利用して Web サーバにデータを送信する場合に，送信前に HTTP リクエストボディを取得し，Android 端末の内部ストレージに保存する（図 5 の（処理 3））．

Web アクセスの応答処理を行うメソッドにより HTTP レスポンスヘッダの受信完了後に HTTP レスポンスヘッダを取得し，Android 端末の内部ストレージに保存する（図 5 の（処理 4））．ただし，HTTP レスポンスヘッダを取得する際，HTTP レスポンスヘッダのオフセットを算出する必要がある．これは，Web サーバから HTTP レスポンスヘッダを受信する際に，HTTP レスポンスヘッダと HTTP レスポンスボディの一部を受信しているためである．HTTP レスポンスヘッダのオフセットは，FindAndParseResponseHeaders() メソッドにより算出する．

HTTP レスポンスボディは，そのサイズが大きい場合，数回に分割されて Web サーバから送信される．このため，各 HTTP レスポンスボディの受信完了後に HTTP レスポンスボディを取得し，Android 端末の内部ストレージに保存する（図 5 の（処理 5））．

一方，(2)SocketPosix クラスは，Web アクセスを行うためのシステムコールを発行する．提案機構は，SocketPosix クラスの Connect() メソッド内で発行された connect() システムコールの戻り値を取得し，Android 端末の内部ストレージに保存する（図 5 の（処理 1））．これにより，ソケット接続の際のコネクションエラーを取得する．ただし，WebView を介した Web アクセスは，非同期処理で行われるため，connect() システムコール発行直後の返り値は，Operation now in progress というエラーが返ってくる．このため，connect() システムコールの処理が完了した直後に，再度 connect() システムコールの戻り値を取得する必要がある．再度 connect() システムコールの返り値を取得するためには，ConnectCompleted() メソッドを使用する．

4.2 提案機構を適用した WebView の導入方法

既存の WebView は，Android 端末にシステムアプリケーションとしてインストールされている．ここで，提案機構を適用した WebView を Android 端末に導入する場合，既存の WebView をアンインストールする必要がある．Android 端末にインストールされているシステムアプリケーションをアンインストールする方法として，Android 端末の root 化がある．Android 端末の root 化とは，一般ユーザから管理者権限ユーザにユーザを変更できるようにする仕組みである．これにより，Android 端末にインストールされている既存の WebView をアンインストールすることができるようになる．

また，Android 端末に提案機構を適用した WebView をインストールする際は，その WebView のパッケージ名を com.google.android.webview とする必要がある．これは，WebView として使用する Android アプリのパッケージ名

表 2　Android 端末の (Nexus 6P) の評価環境

OS	Android 6.0.1
CPU	Snapdragon 810 2.0 GHz (octa core)
メモリ	3 GB

表 3　評価環境

OS	Ubuntu 16.04 LTS
CPU	Intel(R) Xeon E5-2609V4(8 コア)
メモリ	64GB
カーネル	Linux 4.4.0-92-generic (64bit)
Android Emulator	Android 6.0

が決められているためである．

4.3　評価内容と評価環境

提案機構の有用性と提案機構の適用によって生じるオーバヘッドを明確にするために，以下の評価を行った．

（評価 1）提案機構の動作確認

tcpdump を用いて取得する情報と提案機構で取得する情報を比較し，提案機構によって WebView を介した Web アクセスを観測できているか否か検証する．また，その比較結果から提案機構の有用性を示す．

（評価 2）提案機構に要する処理時間の測定

提案機構を適用した WebView を Android 端末に導入し，提案機構によって生じるオーバヘッドを測定する．ただし，通信回線の通信速度によって評価結果に差が生じる可能性が高いため，提案機構に要する処理時間を測定する際は，通信処理部分を含まないこととする．このため，提案機構各処理部分のみの時間を測定する．

また，評価環境を**表 2** と**表 3** に示す．

評価には，自作したテストアプリを使用した．このアプリは，WebView を利用し，Android アプリ内に岡山大学のホームページを表示するアプリである．

4.4　提案機構の動作確認

提案機構で取得した情報とパケットキャプチャツールで取得した情報と比較することで，提案機構で WebView を介した Web アクセスを観測できているか否か検証する．また，その比較結果から，提案機構の有用性を示す．なお，本評価は，Android Emulator を用いて行い，また，パケットキャプチャツールは tcpdump を用いる．

本評価ではテストアプリのみを起動し，tcpdump を用いて収集した通信ログからテストアプリの通信ログを抽出した．なお，テストアプリの通信ログの抽出には，Wireshark の検索機能を利用して，テストアプリの HTTP リクエストに含まれている X-Requested-With ヘッダを検索した．

表 4 に提案機構で取得した通信ログと tcpdump で取得した通信ログの比較結果を示す．なお，表 4 中の「○」は，

表 4　提案機構で取得した通信ログと tcpdump で取得した通信ログの比較結果

	提案機構	tcpdump
HTTP の通信内容	○	○
HTTPS の通信内容	○	–
タイムスタンプ（リクエスト日時）	○	○
Android アプリのパッケージ名	○	–
Web コンテンツの URL	○	○
通信先の IP アドレス	○	○
Web サーバのポート番号	○	○
ソケット接続の際のコネクションエラー	*	–

取得できた情報であり，「–」は取得できなかった情報である．また，「*」は，テストアプリでは，表 1 の（処理 1）は実行されないため，未確認である．表 4 から，tcpdump で取得した通信ログを提案機構においても取得できていることがわかる．

また，テストアプリにおいて確立したコネクションの数を調査した結果，確立したコネクションの数は 45 個あり，tcpdump では，24 個の HTTP を利用した Web アクセスしか確認することができなかった．一方，提案機構は，tcpdump で取得した 24 個の HTTP を利用した Web アクセス以外に，21 個の HTTPS を利用した Web アクセスを観測できていることを確認した．このことから，提案機構では，HTTPS の通信も観測することができることを示した．さらに，提案機構と tcpdump の違いとして，提案機構は，WebView を利用する Android アプリのパッケージ名や TCP/IP 層の情報を取得できる点がある．

4.5　提案機構に要する処理時間の測定

提案機構の性能を把握するために，テストアプリを 5 回起動し，表 1 に示す処理のうち，HTTP リクエストヘッダ作成時に実行される（処理 2），HTTP レスポンスヘッダ受信完了後に実行される（処理 4），および HTTP レスポンスボディ受信完了後に実行される（処理 5）の 1 つ当たりのコネクションにおける提案機構に要する処理時間を測定し，その平均値を算出した．なお，テストアプリでは，表 1 の（処理 1）と（処理 3）は実行されないため，測定していない．測定結果を**表 5** に示す．

表 5 から，（処理 5）の処理時間は，（処理 2）と（処理 4）の処理時間と比べると大きく，（処理 2）の処理時間は，（処理 4）の処理時間よりも大きいことがわかる．これは，各処理時間は，各処理で取得する情報のデータサイズに依存するためであると推察できる．

（処理 5）において，HTTP レスポンスボディは，数回に分割されて送信されるものもあり，提案機構は，HTTP レスポンスボディの受信毎に（処理 5）を実行し，HTTP レスポンスボディをファイルに出力する．このため，各コネクションにおいて，（処理 5）の実行回数は（処理 2）と（処理 4）の実行回数より多いことがある．以上より，（処

表 5　1 つ当たりのコネクションにおける提案機構に要する処理時間の平均値（単位：ms）

（処理 2）	（処理 4）	（処理 5）
0.357	0.0992	1.76

理 5）の全体の処理時間は，（処理 2）と（処理 4）の処理時間と比べると大きくなる．

また，（処理 2）と（処理 4）で取得する情報のデータサイズについて，（処理 2）で取得する情報のデータサイズの方が大きい．さらに（処理 2）は，Web コンテンツの URL と通信先の IP アドレスを取得するために他クラスのインスタンスのメソッドを利用しているのに対して，（処理 4）は，HTTP レスポンスヘッダを取得するために HTTP レスポンスヘッダのオフセットを算出するメソッドのみを利用している．以上より，（処理 2）と（処理 4）の処理時間について，（処理 2）の処理時間の方が長い．

残された課題として，より多くの Android アプリにおいてオーバヘッドを測定することがある．また，テストアプリ以外の Android アプリにおける提案機構のオーバヘッドは，表 5 に示す測定結果に確立するコネクションの数を乗算することで推測できる．

5. おわりに

WebView を介した Web アクセスを観測するために，Android における WebView の Web アクセス観測機構を提案した．提案機構は，Android 上の Web アクセスのうち，WebView を介した Web アクセスのみを観測し，その通信ログを収集することができ，提案機構を適用した WebView をサンドボックスとして使用することにより，マルウェアや悪意のある Android アプリの挙動を解析できる．また，提案機構は，Android アプリとして実現するため，Android 端末への導入が容易である．さらに，WebView の C++層で Web アクセスを観測するため，HTTPS による暗号化通信の通信内容を暗号化される前に取得することができる．

提案機構の動作確認を行い，tcpdump と同等の情報に加え，HTTPS の通信内容，Android アプリのパッケージ名，および TCP/IP 層の情報を取得できることを示し，提案機構の有用性を示した．性能評価では，1 つ当たりのコネクションにおける提案機構に要する処理時間を測定し，その平均処理時間を算出した．また，評価結果から，各処理に要する処理時間を考察した．

残された課題として，WebView を利用する Android アプリの分析，および WebView を介した Web アクセスの安全性の検証がある．

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] Sui, L.: Strategy Analytics: Android Captures Record 88 Percent Share of Global Smartphone Shipments in Q3 2016, STRATEGY ANALYTICS (online), available from 〈https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2016/11/02/strategy-analytics-android-captures-record-88-percent-share-of-global-smartphone-shipments-in-q3-2016〉 (accessed 2016–12–15).

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

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

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

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

[6] Tuncay, G. S., Demetriou, S., and 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).

[7] 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, ISSN (2013).

[8] Bao, W., Zong, M., and 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, ACM (2017).

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

[10] WebKit: Open source web browser engine, available from 〈https://webkit.org/〉 (accessed 2017-08-27).

[11] The Chromium project: Home of the Chromium Open Source Project, available from 〈https://chromium.org/〉 (accessed 2017-08-27).

[12] 吉田 豊，中村 嘉隆，稲村 浩，高橋 修：ドライブ・バイ・ダウンロード攻撃検知のための悪性サイト情報収集手法の改善，マルチメディア，分散協調とモバイルシンポジウム 2016 論文集，pp.813–818(2016)．