事例で学ぶ
セキュアコーディング

熊谷 裕志
kumagai@ierae.co.jp

自己紹介

- 熊谷 裕志
  - 脆弱性好き
  - ベンチャー企業でアプリ開発、セキュリティ団体で脆弱性の調査研究、セキュアコーディングの啓発活動に従事、その後大手コンサル
  - 現在
    - 株式会社イエラエセキュリティ
      - 脆弱性診断を行っている会社
        - » Web、Android/iOSアプリ、etc…

目次

- はじめに
- 脆弱性診断
- 診断環境
- 事例集
- まとめ

はじめに

脆弱性診断とは

- 脆弱性診断とは？
  - ソフトウエアやシステムに対して脆弱性がないかどうか診断するものである
- 脆弱性診断の目的は？
  - ソフトウエアやシステムにある脆弱性を低減することにある

https://www.ipa.go.jp/files/000032929.pdf

脆弱性とは

- 脆弱性とは？
  - ソフトウエア製品やウェブアプリケーション等において、コンピュータ不正アクセスやコンピュータウイルス等の攻撃により、その機能や性能を損なう原因となり得るセキュリティ上の問題箇所である

https://www.jpcert.or.jp/vh/partnership_guideline2015.pdf

脆弱性診断とは

対象となるソフトウエアやシステムにセキュリティ上の問題箇所があるかどうかを調べること

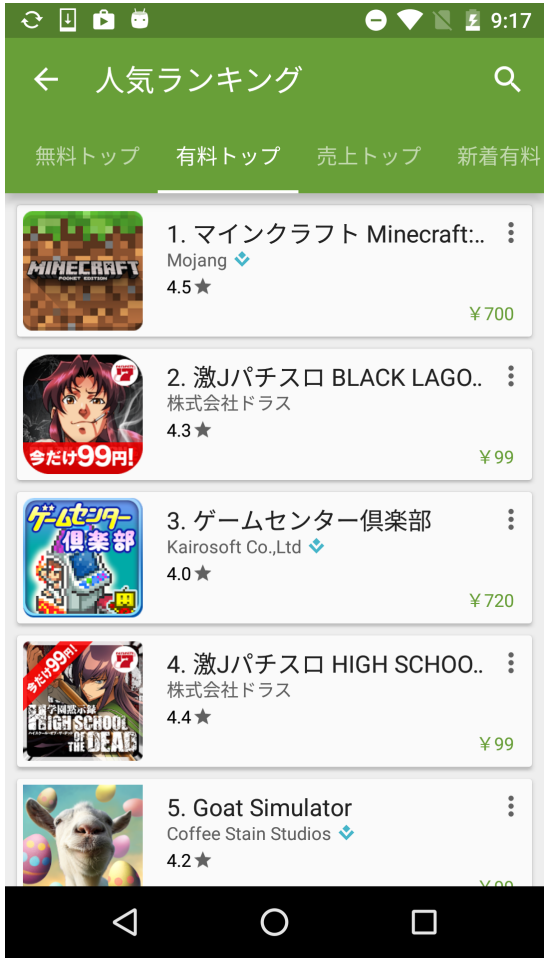
脆弱性診断

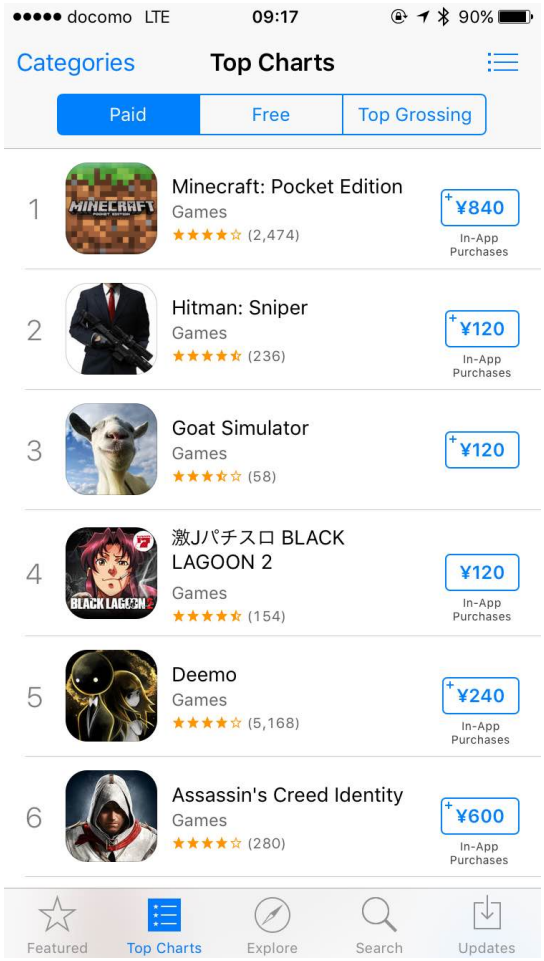
脆弱性診断

脆弱性診断

- スマートフォンアプリの診断
  - Androidアプリの脆弱性診断
  - iOSアプリの脆弱性診断
  - ゲームチート診断

ゲームチート診断





Androidアプリ診断内容

- WebView
  - クロスサイトスクリプティング
  - ローカルファイルアクセス
  - Javascript
- SSL/TLSサーバ証明書検証不備
- デバッグログ
- SDカード
- データ共有機能アクセス制御不備
- 端末データ内の不備
- ファイルパーミッション
- アプリ連携周り等々

iOSアプリ診断内容

- 基本はAndroidと同様
  - AndroidはあるがiOSにはないもの
    - SDカード等対象項目外

ゲームチート診断

- ゲームを優位に進めたりバランスを崩したりすることができてしまう問題
  - 課金回避
  - レアアイテムの取得
  - ステータス、スコア、所有データ改ざん
  - 制約の回避
  - なりすまし等...

診断環境

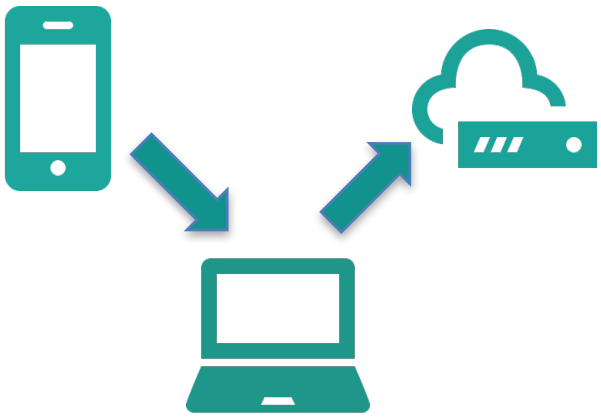
診断環境

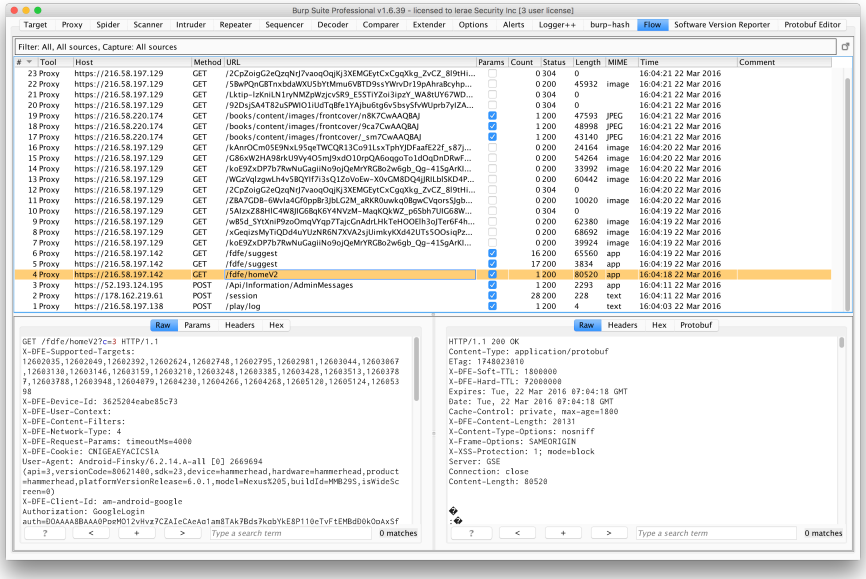
- Burp Suite
  - プロキシ
- JEB
  - Androidアプリ デコンパイラ
- Apktool
  - apkファイルデコード・エンコード
- IDA
  - デバッガ
- ILSpy
  - .NETアセンブリ デコンパイラ

Burp Suite

https://portswigger.net/burp/

- 端末とサーバ間の通信内容を確認するために使用
  - root化した端末
    - iptables等を使用して端末の通信をburpに飛ばす

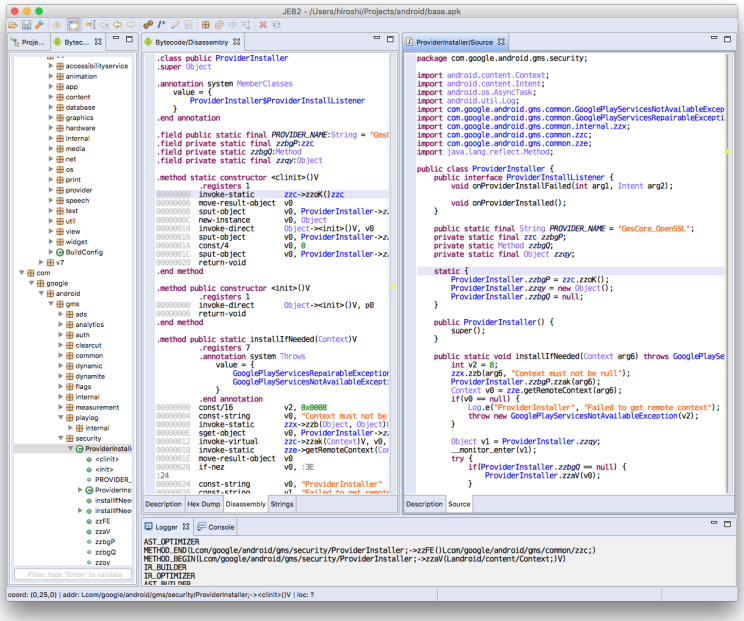




JEB

https://www.pnfsoftware.com/

- Androidアプリ デコンパイラ
  - apkファイルの静的解析のために使用
  - APIが公開されている
  - JavaやPythonでextensionを作成可能



Apktool

http://ibotpeaches.github.io/Apktool/

- apkファイルをデコードしたり再ビルドするために使用

apkファイルをデコードする

```
$ apktool d base.apk
```

```
~/Projects/android/apk ❯❯❯ apktool d base.apk
Picked up _JAVA_OPTIONS: -Dfile.encoding=UTF-8
I: Using Apktool 2.0.3 on base.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /Users/hiroshi/Library/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
~/Projects/android/apk ❯❯❯ ls -la base
total 12
drwxrwxr-x   8 hiroshi staff  272  3 22 16:35 ./
drwxrwxr-x   4 hiroshi staff  136  3 22 16:35 ../
-rw-rw-r--   1 hiroshi staff 4667  3 22 16:35 AndroidManifest.xml
-rw-rw-r--   1 hiroshi staff  279  3 22 16:35 apktool.yml
drwxrwxr-x   4 hiroshi staff  136  3 22 16:35 assets/
drwxrwxr-x   4 hiroshi staff  136  3 22 16:35 original/
drwxrwxr-x 135 hiroshi staff 4590  3 22 16:35 res/
drwxrwxr-x   5 hiroshi staff  170  3 22 16:35 smali/
```

再ビルドする

```
$ apktool b -o base.apk
```

署名する

```
$ jarsigner -verbose -keystore ~/.android/vulnanalysis.keystore
debug.apk
```

IDA

https://www.hex-rays.com/products/ida/

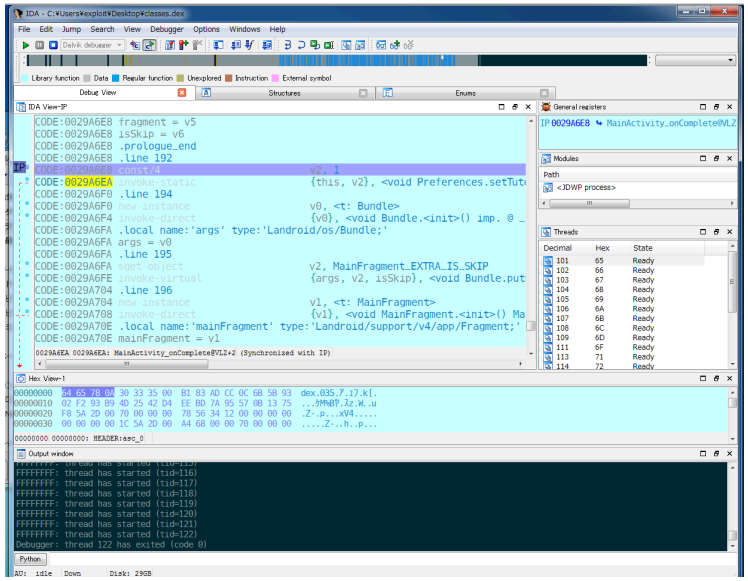
- Androidアプリの動的解析をするために使用

IDAに付属しているandroid_serverを端末で実行

```
$ su
# android_server -p23946
```

ポートフォワードする

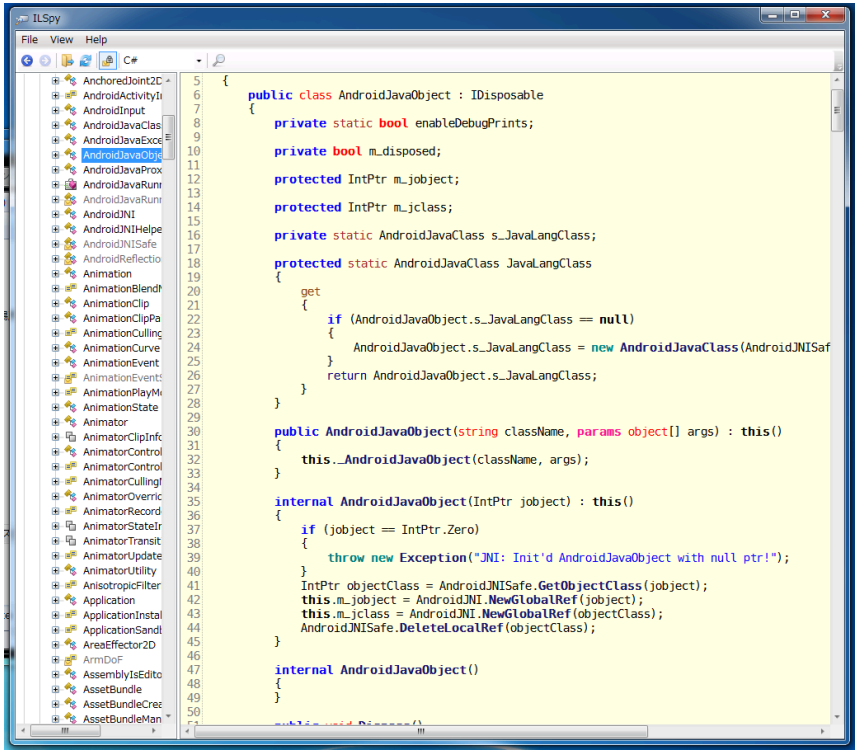
```
> adb forward tcp:23946 tcp:23946
```



ILSpy

http://ilspy.net/

- .NETアセンブリ デコンパイラ
  - .NETアセンブリを静的解析するために使用
  - Unityアプリ
    - 最近のゲーム



事例1

情報漏えい

概要

- 悪意あるアプリにより情報が漏えい
  - WebViewを使用している
  - Intentで受け取ったURLをWebViewに読み込ませる機能がある
  - Intentを受け取るAcivityは公開されている

Intentの値を処理しているコード

1. 最初のActivityでは次のコードが実行される

```
protected void goNextActivity() {
    int v1 = this.getResources().getIdentifier("splash_anim_in", "anim", this.getPackageName());
    int v2 = this.getResources().getIdentifier("splash_anim_out", "anim", this.getPackageName());
    Intent v0 = new Intent(this.getApplication(), this.getNextActivity());
    v0.setFlags(67108864);
    String v3 = GCMIntentService.getUrlExtra(this.getIntent());
    if(StringUtil.isNotBlank(v3)) {
        GCMIntentService.putUrlExtra(v3, v0);
    }

    this.startActivity(v0);
```

2. メインのActivityで次のコードが実行される

```
private boolean openUrlFromNoti(Intent arg5) {
    boolean v0 = false;
    String v1 = GCMIntentService.getUrlExtra(arg5);
    LogUtil.d("[redacted] : " + v1);
    if(StringUtil.isNotBlank(v1)) {
        if(this.webView != null) {
            this.webView.loadUrl(v1);
        }
```

問題を修正したと思われるコード 1/2

次のコードが修正前と後で異なっていた

```
@SuppressLint(value={"SetJavaScriptEnabled"}) private void setWebViewInit(boolean arg13) {
        String v8 = this.getApplicationContext().getDir("localstorage", 0).getPath();
        WebSettings v9 = this.webView.getSettings();
        if(this.isUseBasicAuth()) {
            this.webView.setHttpAuthUsernamePassword(GameSetting.GAME_DOMAIN, "*", GameSetting.BASICAUTH_USERNAME, Ga
              [redacted].oAuth().setBasicAuthorize(GameSetting.GAME_DOMAIN, 80, "*", GameSetting.BASICAUTH_USERNAME, GameS
        }

        v9.setJavaScriptEnabled(true);
        v9.setPluginsEnabled(true);
        v9.setUseWideViewPort(true);
        v9.setLoadWithOverviewMode(true);
        v9.setDomStorageEnabled(true);
        v9.setDatabaseEnabled(true);
        v9.setDatabasePath(v8);
        v9.setAppCacheEnabled(true);
        v9.setAppCacheMaxSize(8388608);
        v9.setUserAgentString(this.userAgent);
        v9.setSaveFormData(true);
        v9.setSavePassword(true);
        v9.setAllowFileAccess(false);
        this.webView.setInitialScale(1);
```

問題を修正したと思われるコード 2/2

ローカルファイルにアクセス出来ないようにした

```
@SuppressLint(value={"SetJavaScriptEnabled"}) private void setWebViewInit(boolean arg13) {
        String v8 = this.getApplicationContext().getDir("localstorage", 0).getPath();
        WebSettings v9 = this.webView.getSettings();
        if(this.isUseBasicAuth()) {
            this.webView.setHttpAuthUsernamePassword(GameSetting.GAME_DOMAIN, "*", GameSetting.BASICAUTH_USERNAME, Ga
              [redacted].oAuth().setBasicAuthorize(GameSetting.GAME_DOMAIN, 80, "*", GameSetting.BASICAUTH_USERNAME, GameS
        }

        v9.setJavaScriptEnabled(true);
        v9.setPluginsEnabled(true);
        v9.setUseWideViewPort(true);
        v9.setLoadWithOverviewMode(true);
        v9.setDomStorageEnabled(true);
        v9.setDatabaseEnabled(true);
        v9.setDatabasePath(v8);
        v9.setAppCacheEnabled(true);
        v9.setAppCacheMaxSize(8388608);
        v9.setUserAgentString(this.userAgent);
        v9.setSaveFormData(true);
        v9.setSavePassword(true);
        v9.setAllowFileAccess(false);
        this.webView.setInitialScale(1);
```

事例2

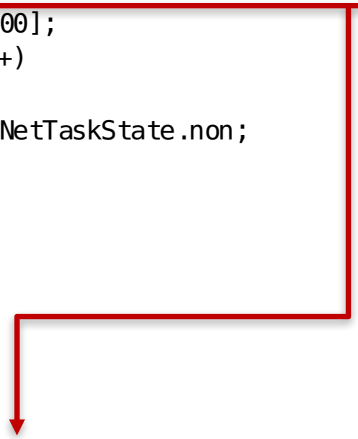
SSL/TLSサーバ証明書検証不備

- SSL/TLSサーバ証明書検証不備 その1
  - Unityで作られているゲーム
  - API通信でサーバとデータをやり取り
  - 通信部分は.NETで実装されている
  - サーバ証明書検証を無視している

SSL/TLSサーバ証明書検証のコード

1. API通信を行う際に次のコードが呼ばれる

```
private void update_netStart()
{
    ServicePointManager.set_ServerCertificateValidationCallback(new
RemoteCertificateValidationCallback(this.OnRemoteCertificateValidationCallback));
    this.netTaskList = new NetManager.NetTaskData[100];
    for (int i = 0; i < this.netTaskList.Length; i++)
    {
        this.netTaskList[i].state = NetManager.enumNetTaskState.non;
    }
    this.listKey_Set = 0;
    this.listKey_Go = 0;
    this.netPhase = NetManager.enumNetPhase.ready;
}
```



2. 独自のコールバックメソッドを作成している

```
private bool OnRemoteCertificateValidationCallback(object sender, X509Certificate certificate, X509Chain
chain, SslPolicyErrors sslPolicyErrors)
{
    return true;
}
```

- SSL/TLSサーバ証明書検証不備 その2
  - アプリのアップデート確認をAPI通信で行う
  - アップデートがあればGoogle Playに誘導
  - サーバ証明書検証を無視している
  - サードパーティ製のライブラリ

- SSL/TLSサーバ証明書検証不備 その2
  - アプリのアップデート確認をAPI通信で行う
  - アップデートがあればGoogle Playに誘導
  - サーバ証明書検証を無視している
  - サードパーティ製のライブラリ

SSL/TLSサーバ証明書検証のコード

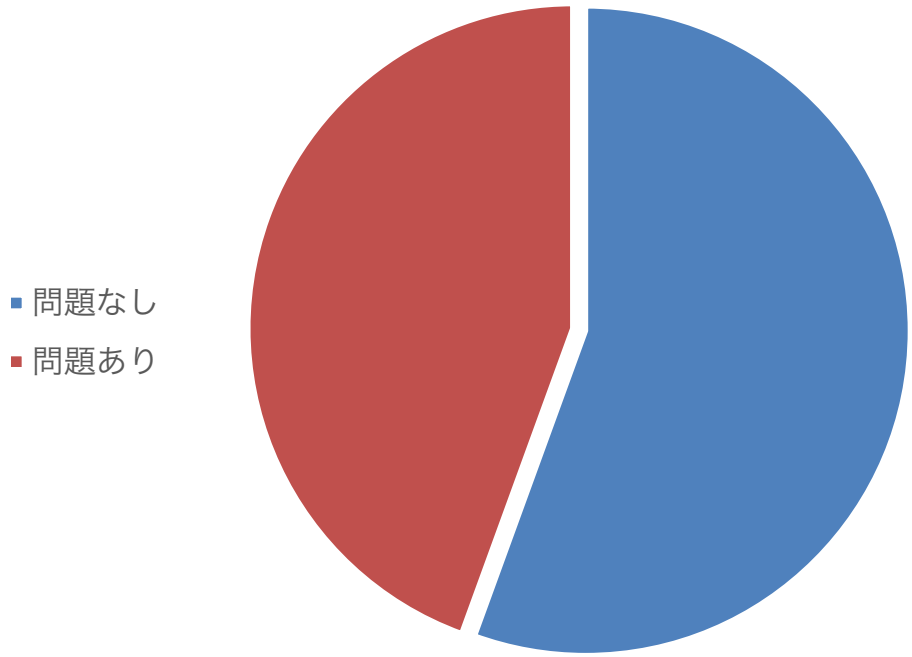
1. アプリのアップデート確認の際に次のコードが呼ばれる

```
private static HttpURLConnection getHttpsConnection(String arg7) throws Exception {
        URLConnection v0;
        KeyManager[] v6 = null;
        URL v1 = new URL(arg7);
        if("https".equals(v1.getProtocol())) {
            TrustManager[] v3 = new TrustManager[]{new X509TrustManager() {
                public void checkClientTrusted(X509Certificate[] arg1, String arg2) throws CertificateException {
                }

                public void checkServerTrusted(X509Certificate[] arg1, String arg2) throws CertificateException {
                }

                public X509Certificate[] getAcceptedIssuers() {
                    return null;
                }
            }};
            SSLContext v2 = SSLContext.getInstance("SSL");
            v2.init(v6, v3, ((SecureRandom)v6));
            HttpsURLConnection.setDefaultHostnameVerifier(new HostnameVerifier() {
                public boolean verify(String arg2, SSLSession arg3) {
                    return 1;
                }
            });
            v0 = v1.openConnection();
            ((HttpsURLConnection)v0).setSSLSocketFactory(v2.getSocketFactory());
```

- SSL/TLSサーバ証明書検証不備があったアプリの割合



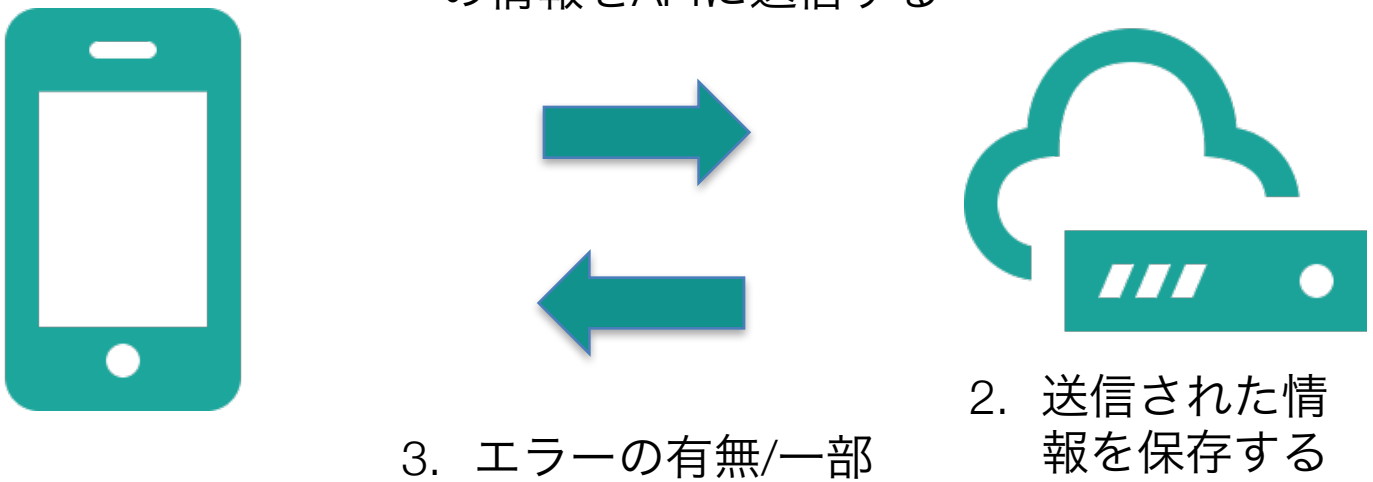
事例3

パラメータ改ざん

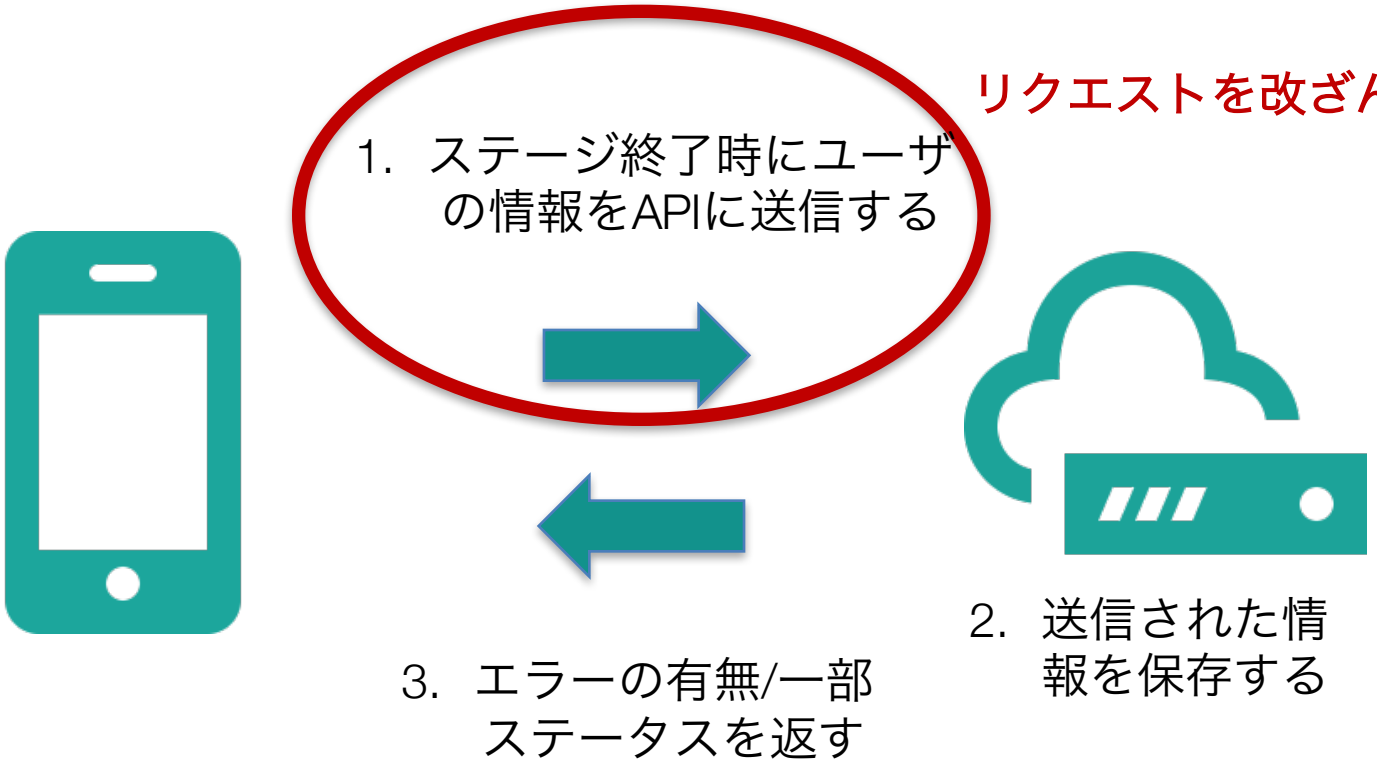
概要

- パラメータを改ざんすることでゲームを優位に進めることが可能
  - スコアを変更できる
  - 課金せずにアイテムを使用できる

スコア変更反映時の処理の流れ

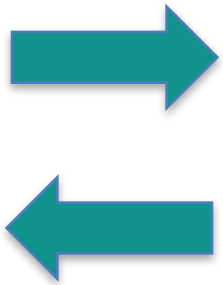


スコア変更反映時の処理の流れ



アイテム使用時の処理の流れ

1. 使用するアイテムの情報と数をAPIに送信する



2. 所有しているか数があるかチェックする

3. OKかNGかを返す

4. OKであれば使用する

アイテム使用時の処理の流れ

1. 使用するアイテムの情報と数をAPIに送信する

2. 所有しているか数があるかチェックする

3. OKかNGかを返す

4. OKであれば使用する

レスポンスを改ざんする

事例3.5

改ざんチェックのバイパス

改ざんチェックのバイパス

- リクエストにハッシュを付加して改ざんを防止している
  - ハッシュは次のものを連結してsha256
    - APIのpath
    - 送信するパラメータ
    - id
  - 簡単にロジックが判明

事例4

root化検知のバイパス

概要

- root化した端末では起動できないように制限をかけているが…
  - わかりやすいクラス名・メソッド名
    - CheckRootedクラス
    - isRootedメソッド
  - これらクラス・メソッドをバイパスするだけで制限を無意味に

事例5

addJavascriptInterface

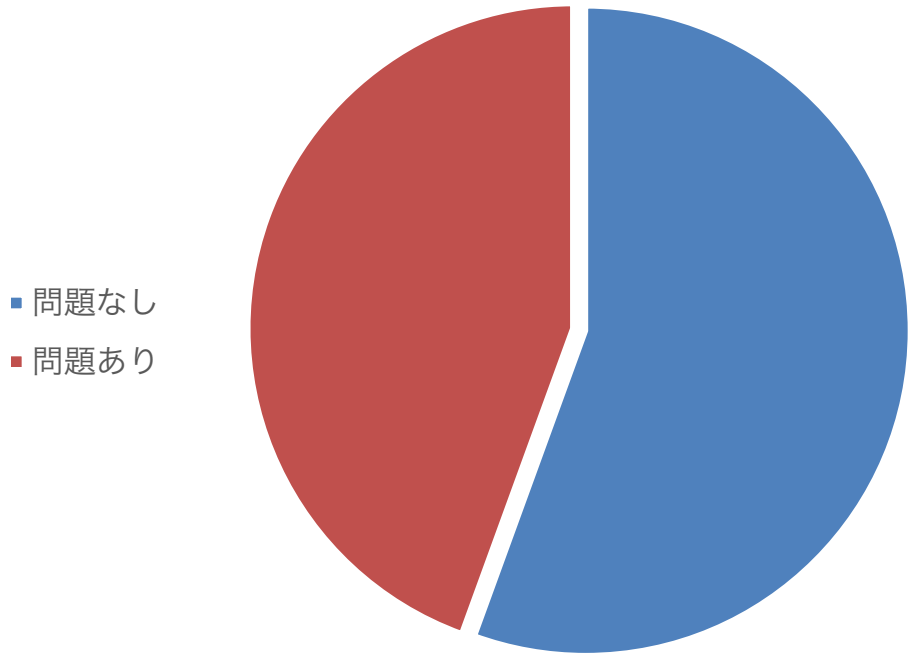
- 今更感のあるaddJavascriptInterface
  - 細工したJavascriptを読みこませることでJavaのコードを実行できる
  - Android 4.2 (API17)以上からはJavascriptInterfaceアノテーションにより機能が制限

```
/** Show a toast from the web page */
@JavascriptInterface
public void showToast(String toast) {
    Toast.makeText(mContext, toast, Toast.LENGTH_SHORT).show();
}
```

http://developer.android.com/guide/webapps/webview.html#BindingJavaScript

現実

- addJavascriptInterfaceの脆弱性があったアプリの割合



まとめ

まとめ

- 未だに作りこむことが多い脆弱性
  - SSL/TLSサーバ証明書検証不備
  - addJavascriptInterface
- JSSECのセキュアコーディングガイドを熟読しよう