サンプル株式会社 御中

サンプルアプリケーション
スマートフォンアプリケーション脆弱性診断報告書

EG セキュアソリューションズ株式会社

2020 年〇月〇日

目次

1. エグゼクティブサマリー

1.1. 総合評価

高危険度（High）

1.2. 総評

脆弱性診断の結果、4 件の脆弱性が発見されました。また、2 件の改善指摘があります。

発見された脆弱性を悪用された場合、各種のチート行為が可能であるため、ゲームバランスが崩れることにより、貴社のサービスを根底から覆される恐れがあります。本書で危険度「High」および「Medium」と表記された問題に対しては、最優先で対策を実施してください。

また、その他、軽微な問題も含めて全般的にアプリケーションに対する保護が十分に考慮されていないと見受けられる設計がされておりました。プログラムを保護していないと不測の事態を招く危険性が高まるため、開発ガイドライン等が整備されていないようでしたら、これらの構築を視野に入れた恒久的な対策の検討をお勧めいたします。

・・・サンプルのため、以下省略・・・

1.3. 内在するリスク

1.3.1. 情報漏洩

指摘事項	危険性
Root 化チェックの不備	High
Javascript による Android メソッドの不正利用	High
重要情報ファイルの共有設定	Medium
暗号化されていないデータベースファイルの存在	Low

1.3.2. 改ざん

指摘事項	危険性
セーブデータ改ざんによるデバッグアイテムの取得	Medium

1.3.3. サービス妨害

指摘事項	危険性
Root 化チェックの不備	High
セーブデータ改ざんによるデバッグアイテムの取得	Medium

※指摘事項ならびに危険度については後述する『2.診断結果』を参照の事。

1.4. 対策指針

1.4.1. 早急の対策

発見された脆弱性の多くは、プログラムの不備(バグ)に起因しています。このため、該当箇所において適切なプログラム改修を行うことで、一般的なセキュリティ水準に引き上げが可能と考えられます。

1.4.2. 恒久的な対策

今回指摘されたような脆弱性がなぜ混入したか、その根本原因を分析する必要があります。今回の診断において検出された問題は、開発者のセキュリティ知識の不足から発生していると推測できます。また本来はテスト工程で検出すべき問題も散見されます。

開発者の知識不足については開発ガイドライン類の整備をする、あるいは開発後のテスト不足などに対してはチェックシートを導入するなど、複合的に予防処置をとることが大切です。

なお、自社内でそのような体制を構築するのが難しい場合は、セキュリティの専門会社にセキュアな開発の構築や運用を委託するなども有効な手段です。様々な角度から抜本的な問題解決に向けた取り組みを実施していくことを推奨いたします。

2. 診断結果

2.1. 指摘一覧

危険度	指摘事項	対応	修正難易度
High	Root 化チェックの不備	プログラム修正	☆☆
High	Javascript による Android メソッドの不正利用	プログラム修正	☆☆
Medium	重要情報ファイルの共有設定	プログラム修正	☆
Medium	セーブデータ改ざんによるデバッグアイテムの取得	プログラム修正	☆
Low	暗号化されていないデータベースファイルの存在	プログラム修正	☆☆
Information	脆弱性の存在する Assets の利用	プログラム修正	☆☆☆
Information	不要なパーミッション設定	プログラム修正	☆

※ 修正難易度 5 段階評価

2.2. 危険度の評価基準

危険度	説明
Critical	High レベルの危険度を保持し、かつ具体的な攻撃手順が発見された脆弱性です。
High	情報漏洩やシステムの停止を招く可能性のある脆弱性です。
Medium	複数の組み合わせにより実害に至る可能性のある脆弱性です。
Low	被害を拡大させる潜在的な脆弱性です。
Information	脆弱性ではありませんが、セキュリティ上および開発上の改善点です。

3. 詳細

3.1. Root 化チェックの不備

概要	
危険度	High
攻撃難易度	容易

説明

該当アプリケーションにおいて root 権限の有無をチェックする処理の保護に不備があるため、チート行為を行うことが可能です。

検証例

① apk ファイルを jar 形式にデコンパイルして復元されたソースコードを閲覧します。この際、ソースコードは ProGuard で難読化を施されていますが、Activity クラスから派生したクラスやそのメソッドは対象外であるため、起動時の onCreate メソッドで root 化等のチェックを行っていることが分かります。

```
protected void onCreate(Bundle paramBundle)
{
  if ((sg.vantagepoint.a.c.a()) || (sg.vantagepoint.a.c.b()) || (sg.vantagepoint.a.c.c())) {
    a("Root detected!");
  }
  if (sg.vantagepoint.a.b.a(getApplicationContext())) {
    a("App is debuggable!");
  }
  super.onCreate(paramBundle);
  setContentView(2130903040);
}
```

② デコンパイルした jar ファイルから java ソースコードや必要なリソースを復元した後に、該当箇所の root チェック処理をコメントアウトしてリコンパイルします。

```
protected void onCreate(Bundle paramBundle)
{
  /*
  if ((sg.vantagepoint.a.c.a()) || (sg.vantagepoint.a.c.b()) || (sg.vantagepoint.a.c.c())) {
    a("Root detected!");
  }
  if (sg.vantagepoint.a.b.a(getApplicationContext())) {
    a("App is debuggable!");
  }
  */
  super.onCreate(paramBundle);
  setContentView(2130903040);
}
```

③ ビルドされた apk ファイルに署名をし直して実行することでチェックを無効にして起動することができます。

影響

本来では不可能な操作やメモリ上にロードされた値の改ざん、ローカル環境のデータの改ざんなどの様々な事象を引き起こされる危険性があります。

対策方法

以下のうちのいずれかの方法で対策が可能です。

■ 対策 1（ネイティブコードの利用）

重要な処理は C や C++言語で記述します。その上で実行形式にコンパイルされたネイティブコードを Java プラグラムから呼び出します。重要な処理を行う箇所はネイティブコードで記述されているため、デコンパイルしてもアセンブラ言語までしか復元できなくなります。このため、リバースエンジニアリングに対して一定の効果が見込めます。

■ 対策 2（有料の難読化ツールの利用）

Activity クラスから派生したクラスやそのメソッドも難読化することが可能な DexGuard や DexProtector などの有料版の難読化ツールを利用します。

該当箇所
以下のアプリケーションが該当します。 ● サンプル Android アプリケーション ➢ XXXXXXXXXXXX.apk

3.2. Javascript による Android メソッドの不正利用

概要	
危険度	High
攻撃難易度	中

説明

該当アプリケーション内の WebView において addJavascriptInterface が許可されております。このため WebView で読み込まれたページで JavaScript からアプリ内の Java メソッドを使用する事ができます。

検証例

① apk ファイルを jar 形式にデコンパイルして復元されたソースコードを閲覧します。

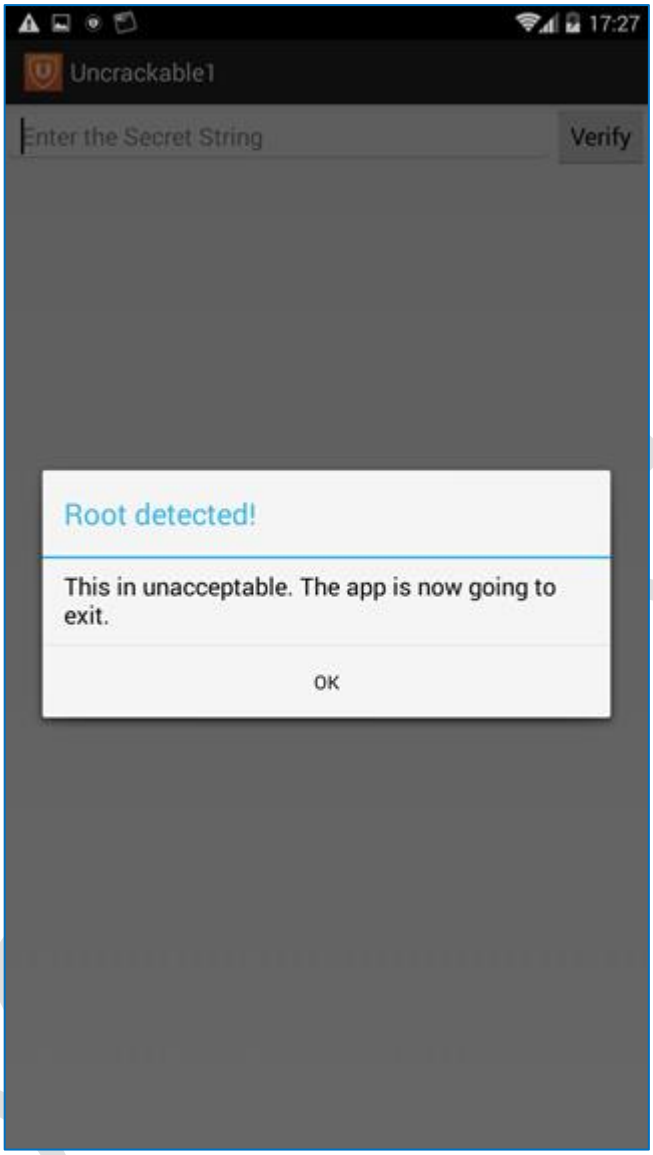
```
super.onCreate(paramBundle);
setContentView(2130903040);
paramBundle = (WebView)findViewById(2131165184);
paramBundle.getSettings().setJavaScriptEnabled(true);
paramBundle.addJavascriptInterface(new WebAppInterface(this), "Android");
```

② DNS 情報を偽装したテスト用 Web サイトの下記の HTML ファイルをアプリケーションから読み込みます。

【偽装用 HTML】

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script type="text/javascript">
    function scriptInterface(){
        Android.XXXXXXXX ();
    }
</script>
<title>Insert title here</title>
</head>
<body>
<input type="button" onclick="scriptInterface (')"/>
</body>
</html>
```

③ 本来の目的とは異なる情報が読み込まれていることが確認できます。



影響

悪意のあるフリーの Wifi スポット等に接続した場合、DNS の名前解決を容易に偽装することができるため、貴システムになりすました罠サイトに接続した利用者の端末内の秘匿情報が漏洩する危険性があります。

対策

addJavascriptInterface を許可して HTML 上から画面を制御する実装は非常に危険です。該当機能が WebView を用いて実装しなければならない機能であるか再検討した上で、可能であれば個別の Activity として実装することを推奨いたします。

該当箇所
以下のアプリケーションが該当します。 ● サンプル Android アプリケーション ➢ XXXXXXXXXXXX.apk

3.3. 重要情報ファイルの共有設定

概要	
危険度	High
攻撃難易度	中

説明

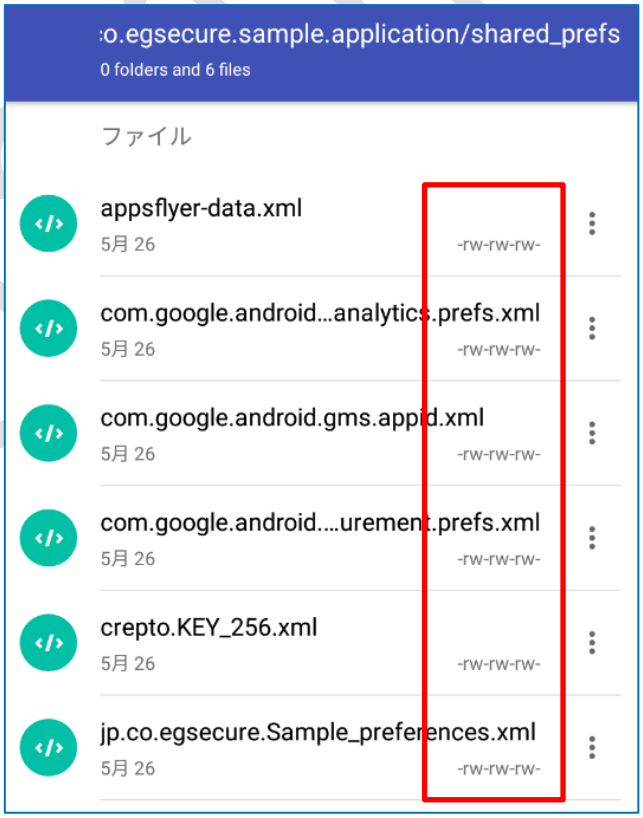
セッション管理に用いられている UUID 情報が他のアプリケーションから読み書き可能です。またこの UUID を暗号化するためのキーも同様のファイルパーミッション設定がなされているため読み書きが可能な状態です。

検証例

下記は、端末内の重要ファイルが格納されているディレクトリー覧を表示した例です。重要情報を格納しているのにもかかわらず、他ユーザー（他のアプリケーション）からの読み書きを許可していることが分かります。

【格納場所】

Android	/data/data/jp.co.egsecure.sample.application/shared_prefs/



また以下は UUID 情報を暗号化したファイルと、そのキーの内容を保存したファイルを表示した例です。

【暗号化された UUID 情報】

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="deviceId">AQLBAq1VTfIRzfUTeFB0kH+0EqVLOH7J3Ae+lqby/
    HWl74LxXJIPvNOeH6/NwwsnrIcUcz670F3BXh+3HAqRYXwu
    </string>
</map>
```

【暗号化キー情報】

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="cipher_key">CR/5f0dwb6qIgyiEMoBtEtSAH8wrLV69diDsLOjdZBQ=
    </string>
</map>
```

これらの情報から暗号化された UUID を復号した上で、セッション ID に変換することが可能です。

影響

悪意のアプリケーションからアクセス可能な状態であるため、ファイル内からセッション情報が漏洩する危険性があります。また、セッション情報が漏洩した場合、なりすましによる重要情報の漏えいへと繋がる恐れがあります。

対策方法

アプリケーションの動作上、必要となる情報や端末上に保存しなければならない重要情報はキーチェーンや Keystore を利用して安全な領域に保存することを推奨いたします。

該当箇所

以下のアプリケーションが該当します。

- サンプル Android アプリケーション
  - XXXXXXXXXXXX.apk

3.4. セーブデータ改ざんによるデバッグアイテムの取得

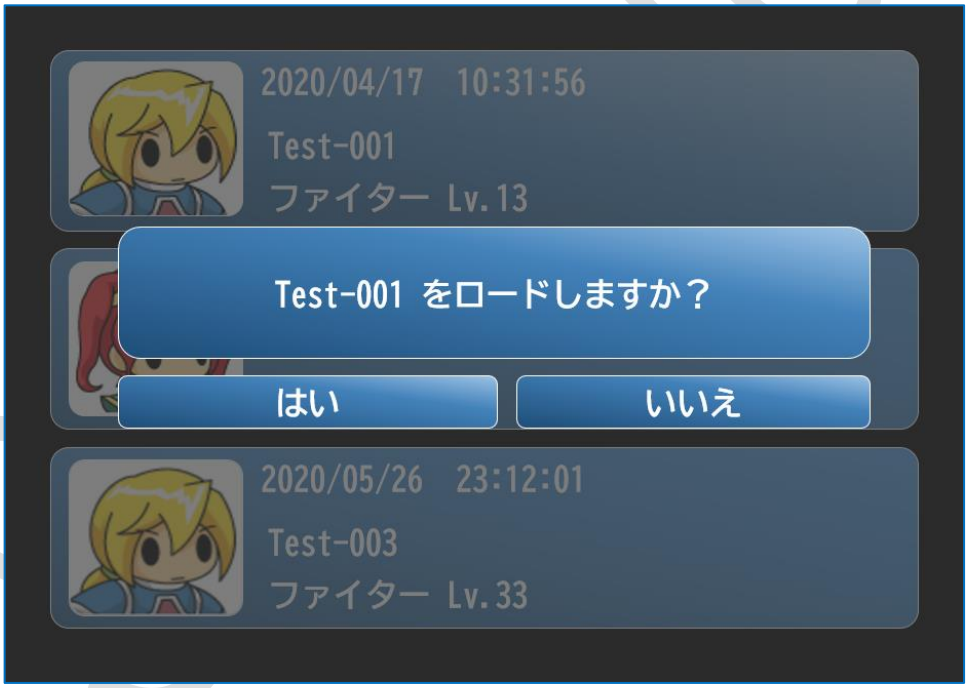
概要	
危険度	Medium
攻撃難易度	容易

説明

端末内にセーブファイルデータが保存されており、このデータのロード時のチェックに不備があるため、特定のアイテムコードを読み込ませることでデバックアイテムを取得することが可能です。

検証例

① 該当アプリケーションを起動して下記の画面まで操作します。

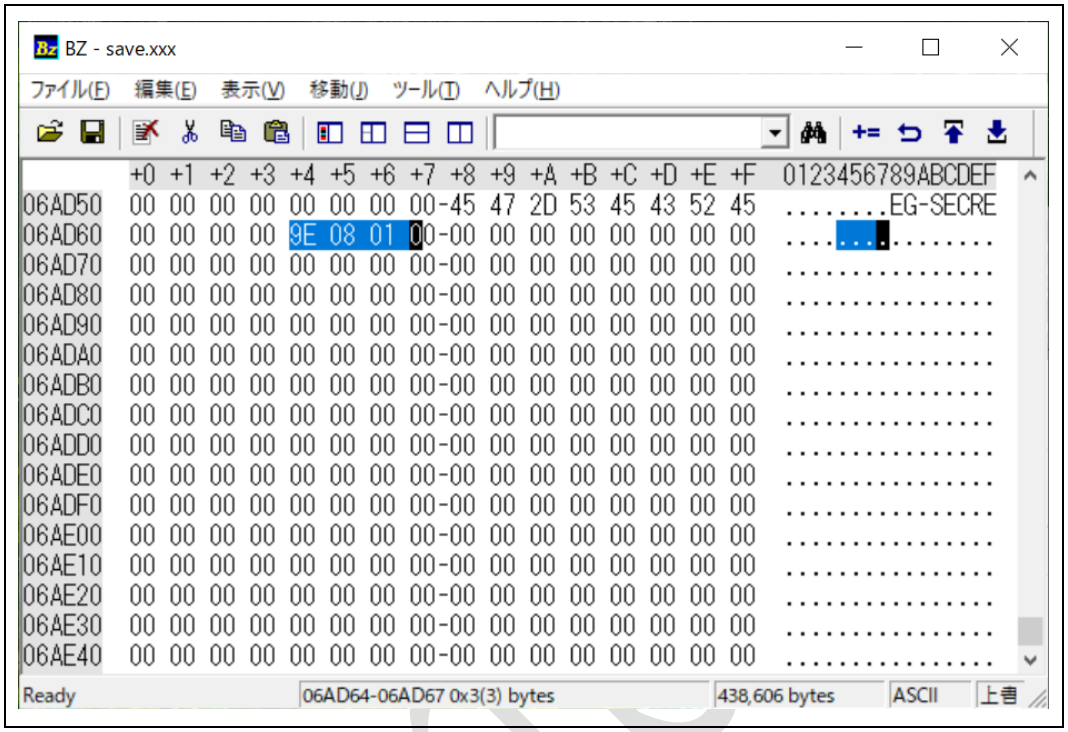


② この状態で端末内の以下のフォルダに格納されているセーブファイル（save.xxx）を PC 上に取り出します。

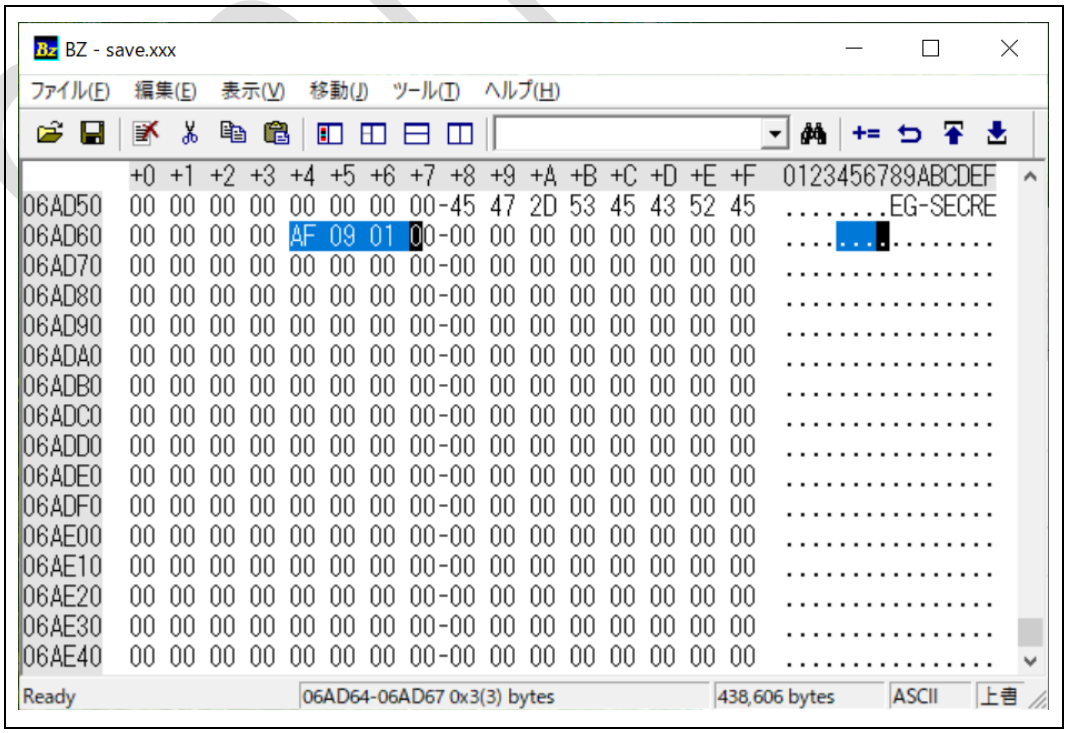
【格納場所】

Android	/data/data/jp.co.egsecure.sample.application/files/
iOS	/var/mobile/Containers/Data/Application/<uuid>/Documents/

③ 取り出したセーブファイル（save.xxx）をバイナリーエディタで開き、アイテム「回復薬」のアイテムコード「0x9E08」を検索します。

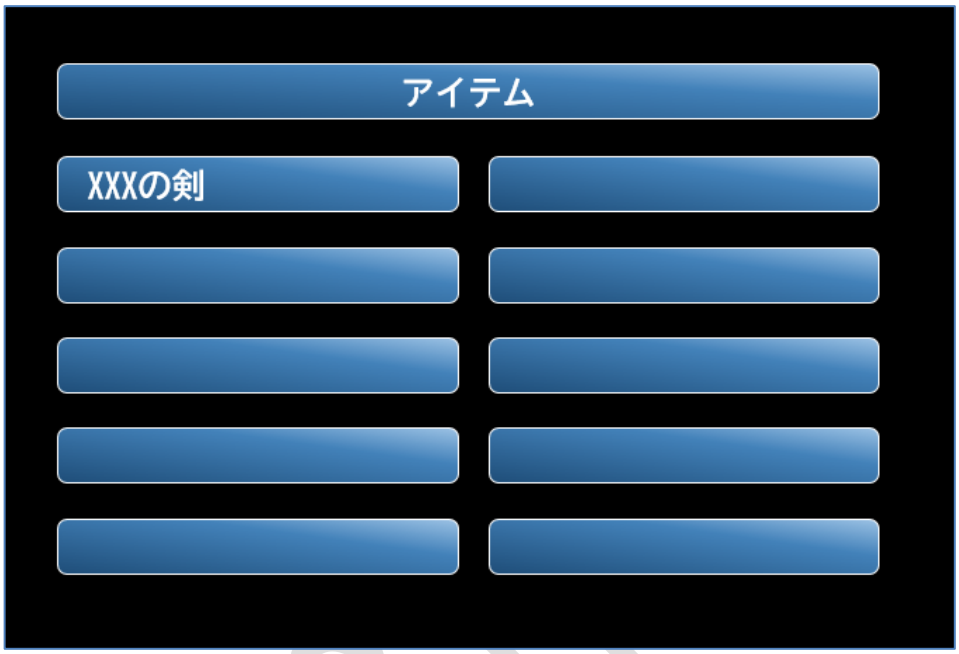


④ 該当の保存領域が見つかった後、デバックアイテムである「XXXX の剣」のアイテムコードである「0x AF09」で上書き保存します。



⑤ 改ざんしたセーブファイル（save.xxx）を端末上の元の格納位置にコピーします。

⑥ そのままゲームを起動するとメモリ上のアイテムデータが改ざんされて下記のようなデバッグ用のアイテムを所持している状態になります。



影響

デバッグの用途に使われるアイテム等は非常に強力な効果を持っているため、チート行為に利用されることになり、ひいてはゲームバランスを崩壊させることに繋がります。

対策方法

以下の対策を両方とも実施することを推奨いたします。

■ 対策 1（デバッグアイテムの無効化）

リリース済みのアプリケーションにおいては、通信先のサーバーおよびアプリケーションの双方においてデバッグアイテムを無効化します。

■ 対策 2（クラウド環境へのセーブデータを保存）

セーブデータ等の重要なデータはサーバー側で保存します。またアプリケーション内でその情報を利用する際には、メモリ上に読み出す際に XOR 処理等を通して容易に検索できないようにした上でメモリにロードすることを推奨します。

該当箇所
以下のアプリケーションが該当します。 ● サンプル Android アプリケーション ➢ XXXXXXXXXXXX.apk ● サンプル iOS アプリケーション ➢ XXXXXXXXXXXX.ipa

3.5. 暗号化されていないデータベースファイルの存在

概要	
危険度	**Low**
攻撃難易度	容易

説明

アプリケーション内で利用している情報が暗号化されずに端末上に保存されております。そのため、この値を改ざんすることで例外処理を発生させることが可能です。

検証例

端末内の以下のフォルダに格納されている sqlite DB ファイル（post.db）を表示したものです。

【格納場所】

Android	/data/data/jp.co.egsecure.sample.application/databases/
iOS	/var/mobile/Containers/Data/Application/<uuid>/Documents/

暗号化されずにデータが格納されていることが確認できます。

影響

端末の紛失時や PC へのデータのバックアップの際に重要情報が漏洩する危険性があります。

対策方法

アプリケーションの動作上、必要となる情報や端末上に保存しなければならない重要情報はキーチェーンを利用して安全な領域に保存することを推奨いたします。

該当箇所
以下のアプリケーションが該当します。 ● サンプル Android アプリケーション ➢ XXXXXXXXXXXX.apk ● サンプル iOS アプリケーション ➢ XXXXXXXXXXXX.ipa

3.6. 脆弱性の存在する Assets の利用

概要	
危険度	**Information**
攻撃難易度	容易

説明

利用している Unity Assets「XXXXXX」のバージョンには、暗号化する際に利用する秘密鍵の管理方法に問題があります。そのため、Assets 内に存在する C#言語でコンパイルされた実行ファイルを逆アセンブリすることで秘密鍵を取得することが可能です。

検証例

【参考情報】

「XXXXXX」http://xxxxxxxxx/xxxxx/xxxxxx/xxxxx

影響

サーバー側の API に脆弱性が存在していないため、現状、問題はありませんが、暗号化して送受信している POST データとレスポンスデータを復号されて中身を閲覧される可能性があります。

対策方法

該当する Unity Assets「XXXXXX」のバージョンを最新バージョンにアップデートすることを推奨いたします。

該当箇所

以下のアプリケーションが該当します。

- サンプル Android アプリケーション
  - XXXXXXXXXXXX.apk
- サンプル iOS アプリケーション
  - XXXXXXXXXXXX.ipa

3.7. 不要なパーミッション設定

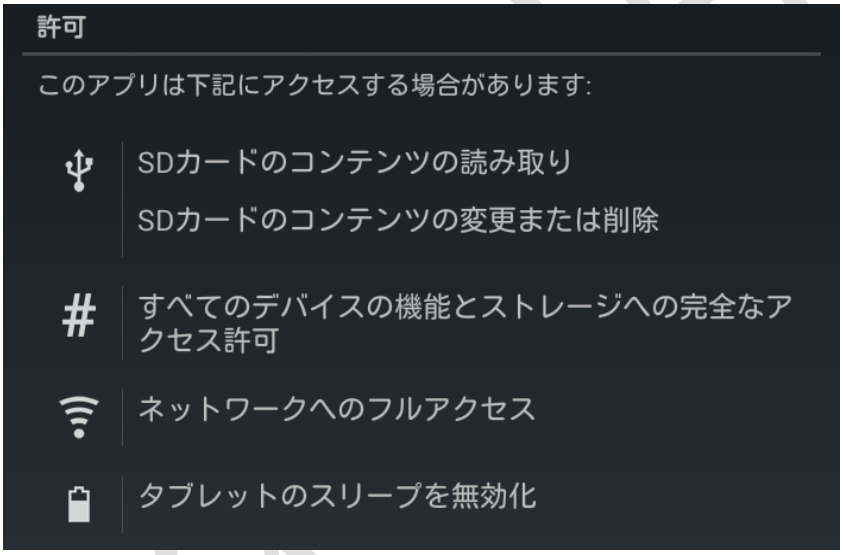
概要	
危険度	Information
攻撃難易度	容易

説明

該当アプリケーションが動作する上で、必要がないと思われるパーミッションを要求しております。

検証例

以下は端末上で取得しているパーミッション情報を表示したものです。



影響

該当アプリケーションに対してのセキュリティ上の脅威はありませんが、利用者のプライバシー保護の観点から好ましい状態ではありません。

対策方法

アプリケーションが動作する上で不要なパーミッションは AndroidManifest.xml から削除することをお勧めいたします。

該当箇所

以下のアプリケーションが該当します。

- サンプル Android アプリケーション
  - XXXXXXXXXXXX.apk

4. 診断概要

4.1. 診断情報

サービス名	サンプルアプリケーション
アプリケーション	サンプル Android アプリケーション サンプル iOS アプリケーション
API	http://example.jp/
診断方法	当社エンジニアによるマニュアル診断
サービス状態	本番稼働前環境
診断実施者	社員 1、社員 2、徳丸　浩
診断期間	2020/1/1～2020/12/31
診断時間	10:00 ～ 18:00

4.2. 診断項目

カテゴリ	診断項目	iOS	Android
データ保護	アカウント情報の保護	○	○
	アプリケーションログ	△	○
	キーボードキャッシュの無効化	○	○
	クリップボードの許可状況	○	○
	パスワードやPINの画面表示	○	○
	OSバックアップファイルへの機密データ出力	△	ー
	共有ファイルの利用状況	○	○
	キャッシュファイルの利用状況	○	○
	最低限のパーミッション設定	○	○
	機密データ利用目的への説明	○	○
	機密データのメモリロードと破棄状況	△	○
暗号化	暗号化キーのハードコーディング	△	○
	脆弱な暗号化方式の採用	△	○
	脆弱な暗号化アルゴリズムの利用	△	△
	暗号化キーの使いまわし	△	△
	乱数ジェネレーターの強度	△	△
認証とセッション管理	認証処理の不備	△	△
	セッションIDの生成方法	○	○
	ログアウト	○	○
	パスワードポリシー	○	○
	不適切なバイオメトリクス認証の実装	△	ー
	ログイン通知機能の不備	○	○
ネットワーク通信	TLSの利用	△	△
	SSL証明書の検証不備	△	△
	証明書ストアの利用方法	△	○
プラットフォームAPI	不要なAPIの使用	※1	○
	データ入力値へのチェック処理	○	○
	カスタムスキームの利用状況	○	○
	危険なスキームの利用	○	○
	WebViewのJavascriptアクセス	ー	○
	危険なIMEIやUUIDの利用方法	△	△
	危険なシリアル化APIの使用	△	△
プログラムコードとビルド設定	不適切なプロビジョニング	○	ー
	デバッグモードの有効化	△	△
	デバッグシンボルの削除状況	△	○
	利用コンポーネントの既知の脆弱性	△	△
	エラーハンドリング状況	○	○
	セキュリティコントロールのエラー設定状況	○	ー
	メモリリークやメモリ保護の状況	△	△
	スタック保護やPIEサポートの設定状況	△	△
リバースエンジニアリング	不適切なRootチェック	○	○
	不適切なエミュレータチェック	○	○
	機密データのメモリへの直接ロード	△	△
	データ改ざん検知機能の有無	○	○
	リバースエンジニアリングツールの検知	○	○
	デバッグプロトコルの許可状況	△	△
	プログラムコードの難読化	○	○
サーバーAPI	※ Webアプリケーション脆弱性診断の診断項目を参照のこと		

5. 免責

5.1. 診断の正確性

脆弱性診断は、テスト用のアプリケーションやサーバーに対して、攻撃者の立場から関連ツールを操作して行なう疑似攻撃テストを実施しております。

脆弱性診断の特性上、本報告書に記載する脆弱性については再現性・網羅性を完全に保証するものではありません。脆弱性への対策指針をもとにプログラム修正その他の対策を行なわれる場合、貴社の責任で行なって頂きますようお願いします。

5.2. 本報告書に関するお問合せ

本報告書に関するご質問・その他お問い合わせは、本報告書のご提出日より起算して 3 ヶ月間承ります。ただし、本報告書をもとにお客様がプログラム修正などの対策を実施される場合については、個別の実装方法についてのご質問にはお答えしかねますことをご了承ください。

期間終了後のお問い合わせについては、別途ご相談ください。

お問合せ先： contact@eg-secure.co.jp