Play Integrity APIを用いたゲームアプリを保護する手法

荒川 湧佑

uname

- Yusuke Arakawa（荒川 湧佑）

nekolaboratory

arakawa.yusuke

know the cheat

ゲームチートの手法はざっくり分類すると以下の３分類

- **メモリハックによるゲーム内パラメータ改変**

  e.g. ゲーム中，プレイヤーのHPを格納するメモリアドレスを特定，改変することで有利な状況を作る

- **通信改竄によるゲーム結果改変やパラメータ改変**

  e.g. ゲーム結果をサーバーに通信する際に該当のエンドポイントへの通信をブロックし，偽装した有利な状況を送信する

- **アプリバイナリ改変によるゲーム挙動改変**

  e.g. アプリバイナリを改変することで，ゲームの挙動を常に有利な状態にする

know the cheat

それぞれのゲームチートの対策方法をざっくり解説

- **メモリハックによるゲーム内パラメータ改変**

  重要な値はメモリ内部に格納する際に難読化を行い，メモリアドレスが割れない工夫をする

- **通信改竄によるゲーム結果改変やパラメータ改変**

  通信内容を暗号化して，改竄検知用パリティをチェックする

- **アプリバイナリ改変によるゲーム挙動改変**

  アプリバイナリに含まれる様々なデータのハッシュをチェックする

know the cheat

それぞれのゲームチートの対策方法をざっくり解説

- **メモリハックによるゲーム内パラメータ改変**

  重要な値はメモリ内部に格納する際に難読化を行い，メモリアドレスが割れない工夫をする

- **通信改竄によるゲーム結果改変やパラメータ改変**

  通信内容を暗号化して，改竄検知用パリティをチェックする

- **アプリバイナリ改変によるゲーム挙動改変**

  アプリバイナリに含まれる様々なデータのハッシュをチェックする

What is "Play Integrity API"?

- Googleから提供されているAPKとAPKの動作環境検証用API
  - Android 4.4以降で動作する
  - SafetyNetの後継 *後述
- 無料で使用できる

What is ”Play Integrity API”?

Play Integrity APIで検証できる要素は以下の３点

- **正規のアプリバイナリ**

  改変されていないバイナリとやり取りしているかどうかをチェックする

- **正規の Play インストール**

  現在のユーザー アカウントにライセンスが付与されているかどうかをチェックする

- **正規の Android デバイス**

  Google Play開発者サービスを搭載した正規のデバイスでアプリが動作しているかどうかをチェックする

と，その前に・・・

前段のSafatyNet APIを紹介

About "SafetyNet Attestation API"

- Googleから提供されているAPKとAPKの動作環境検証用API

  Android 2.3以降で動作する

  無料で使用できる

  デバイスの完全性に関する判定結果（Boolean），APK証明書の証明書ハッシュ（String）

- 2023年6月から段階的に廃止予定（Play Integrity APIへの移行を推奨）

About "SafetyNet Attestation API"

SafetyNet Attestation の廃止スケジュール

日付	マイルストーン	それはあなたにとって何を意味しますか？
2022年6月	発表	できるだけ早く Play Integrity API との統合を開始する必要があります。それまでの間、SafetyNet Attestation はアプリで引き続き機能します。
2023年6月	移行期限	Play Integrity API に移行した場合、SafetyNet Attestation はアプリの以前のバージョンで引き続き機能します。アプリの以前のバージョンとの危険な相互作用を引き続き検出できます。移行していない場合、SafetyNet Attestation はアプリ (以前のバージョンを含む) で機能しなくなり、エラーが返されます。アプリが本番環境で Play Integrity API を呼び出す場合、移行したと見なされます。
2024年6月	フルターンダウン	SafetyNet Attestation は、アプリのどのバージョンでも機能しなくなります。アプリはエラーを受け取ります。

- **https://developer.android.com/training/safetynet/deprecation-timeline**

Improvements from "SafetyNet"

SafetyNet Attestation APIからPlay Integrity APIでの移行で得られるメリット

- **レポートの検証にGoogleサーバが使用できるようになった**

  SafetyNetでは自前でJWTの検証が必要だった

- **各ドメインレベルのレポートが他のレポートへ干渉することがない**

  SafetyNetでは検証環境が要件に満たない場合，正常系と異なるフォーマットのJSONが返却されていた

- **ライセンス認証ができるようになった**

  有料アプリを正規に購入したユーザーかどうかを検証できるようになった

- **Unity用SDKが配布されるようになった**

  これまではUnityに実装する場合，native SDKを実装する必要があった

Improvements from "SafetyNet"

SafetyNet Attestation APIからPlay Integrity APIでの移行で得られるメリット

- **レポートの検証にGoogleサーバが使用できるようになった**

  SafetyNetでは自前でJWTの検証が必要だった

- **各ドメインレベルのレポートが他のレポートへ干渉することがない**

  SafetyNetでは検証環境が要件に満たない場合，正常系と異なるフォーマットのJSONが返却されていた

- **ライセンス認証ができるようになった**

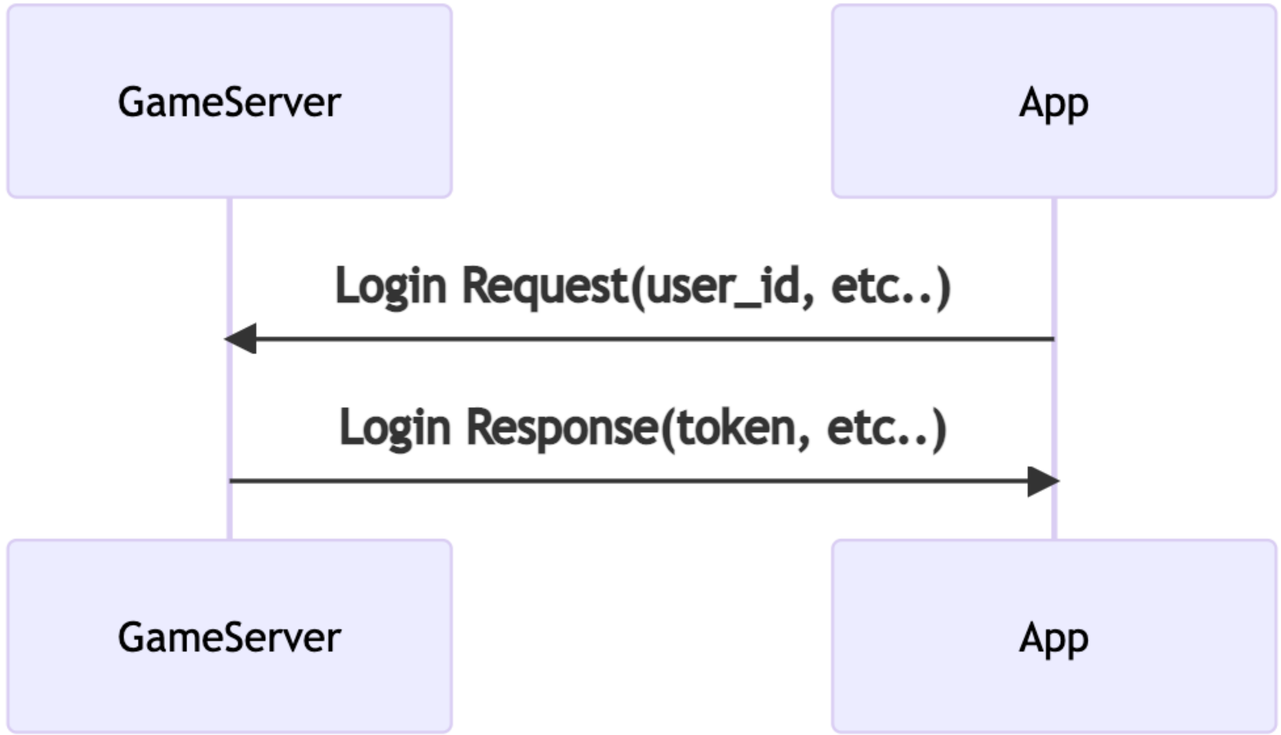
  有料アプリを正規に購入したユーザーかどうかを検証できるようになった

- **Unity用SDKが配布されるようになった**

  これまではUnityに実装する場合，native SDKを実装する必要があった

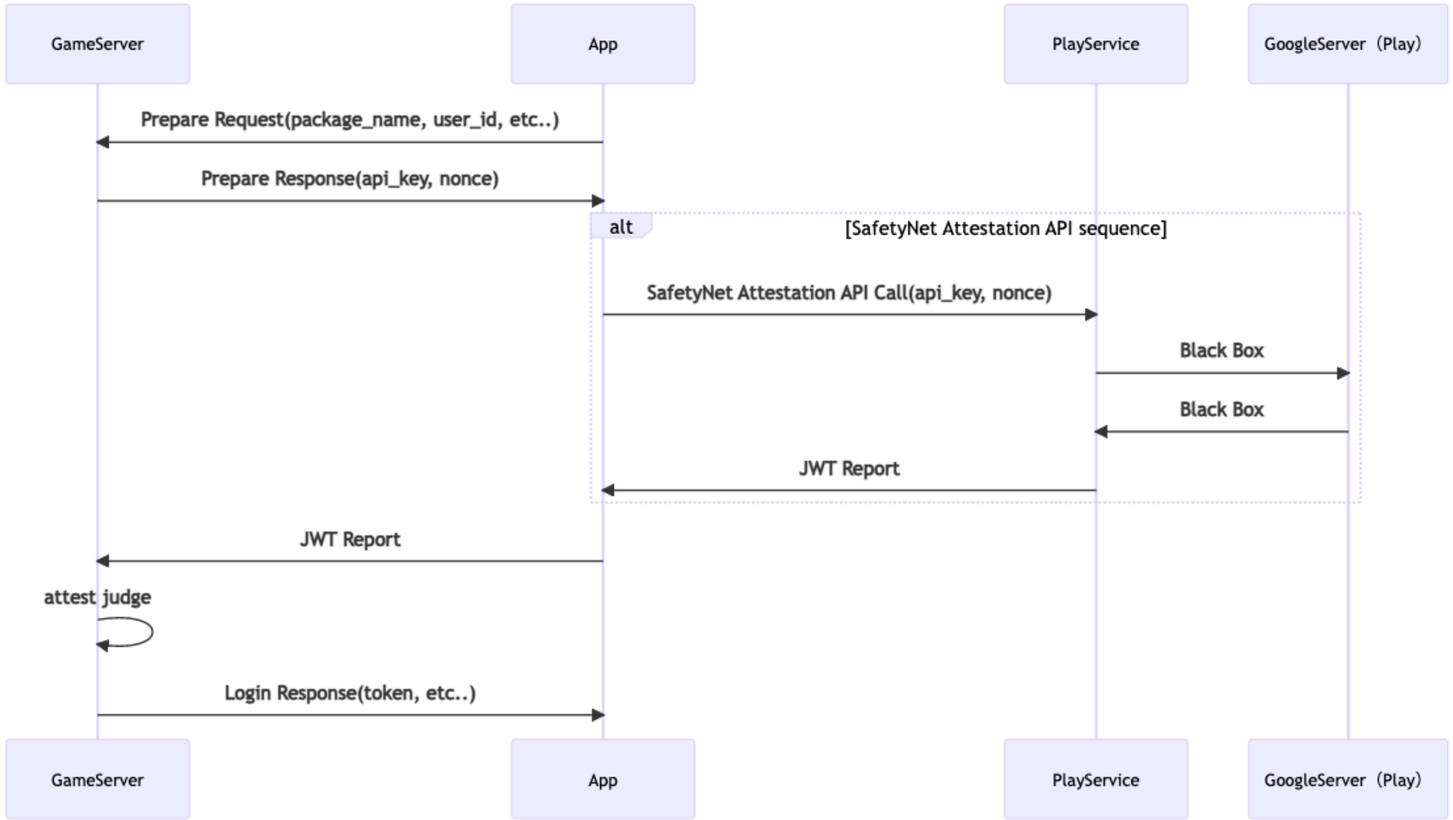
Improvements from "SafetyNet"

未対策のゲームログイン処理フロー



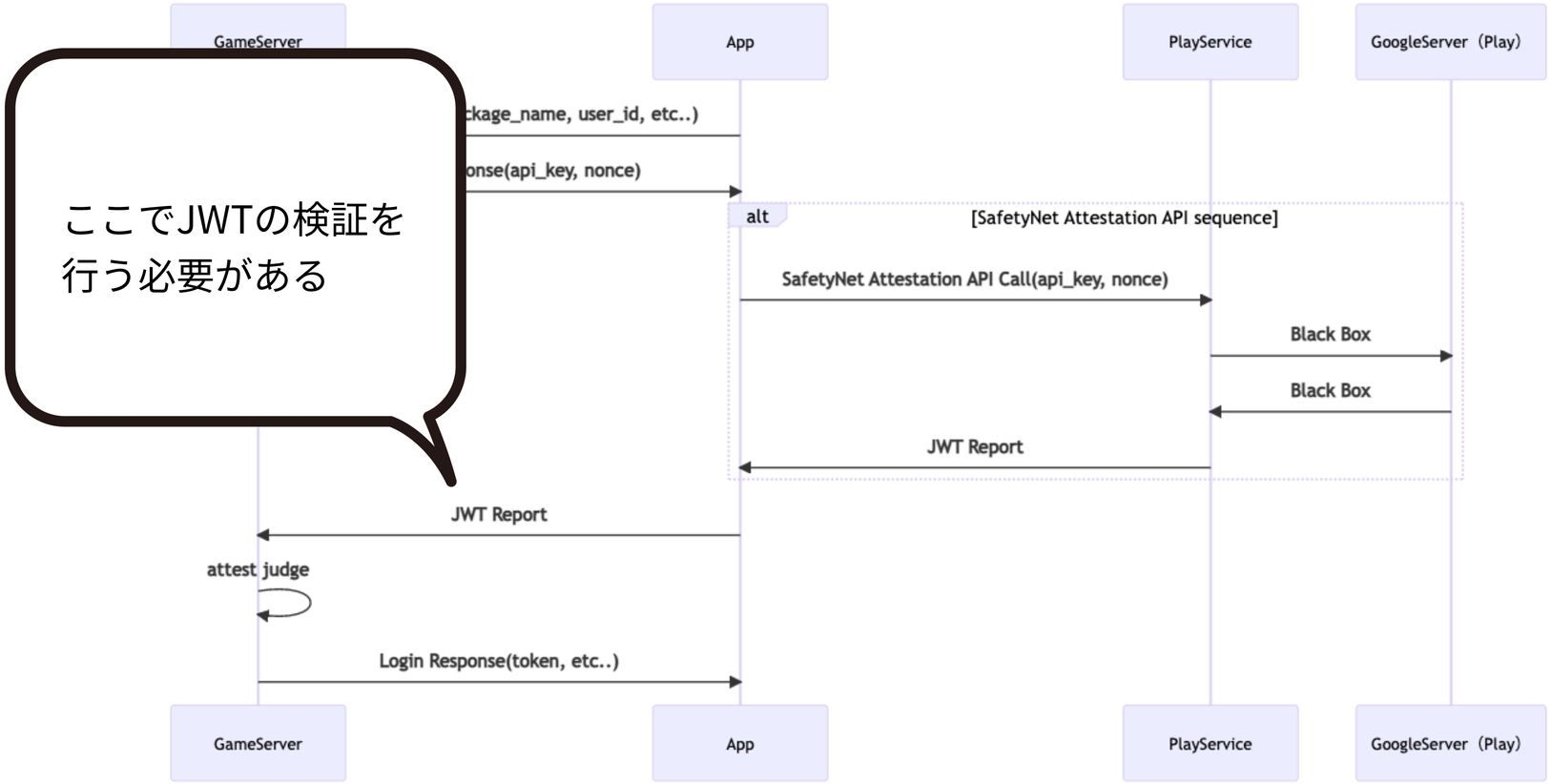
Improvements from "SafetyNet"

SafetyNet Attestation APIの処理フロー



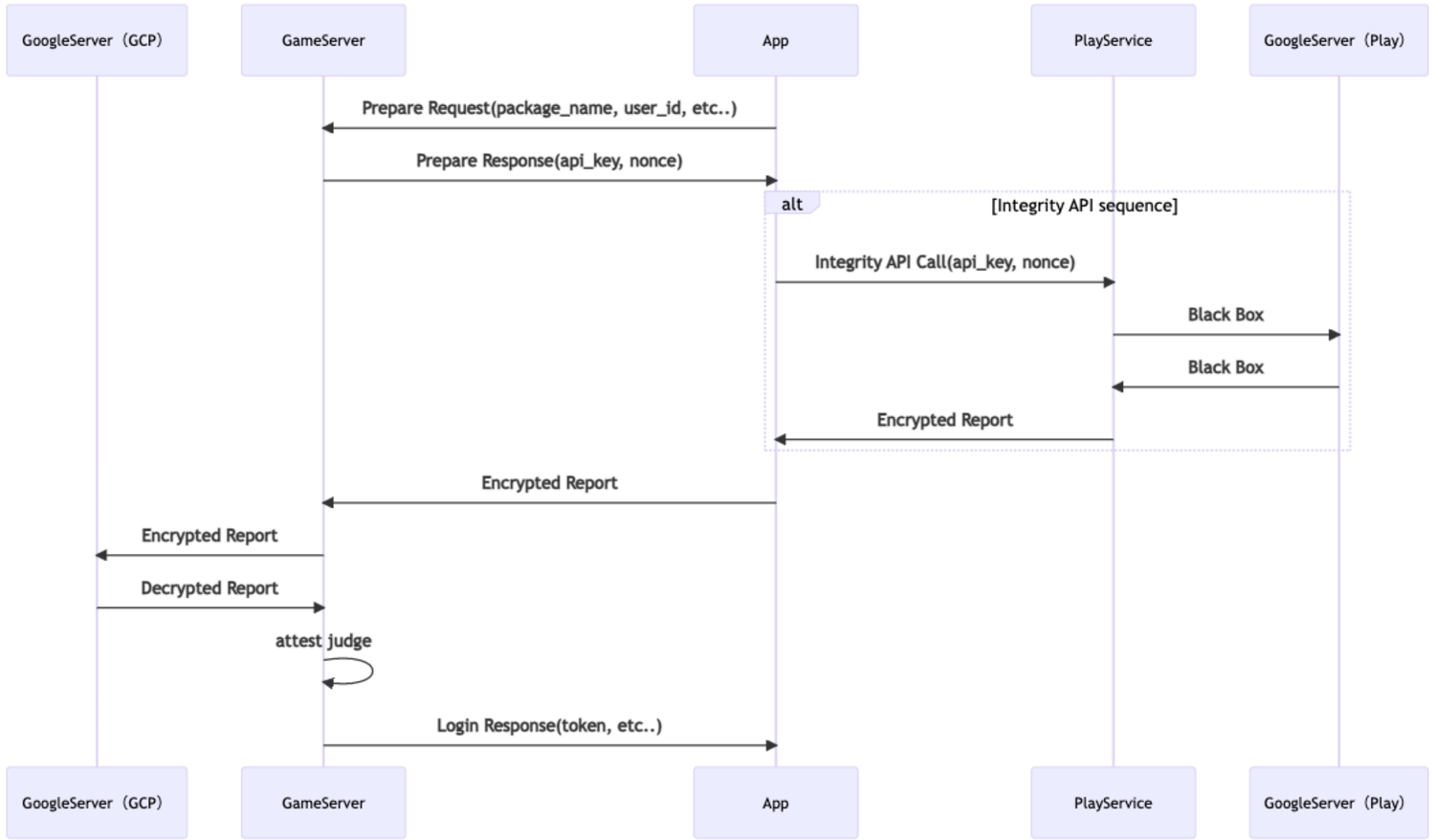
Improvements from "SafetyNet"

SafetyNet Attestation APIの処理フロー



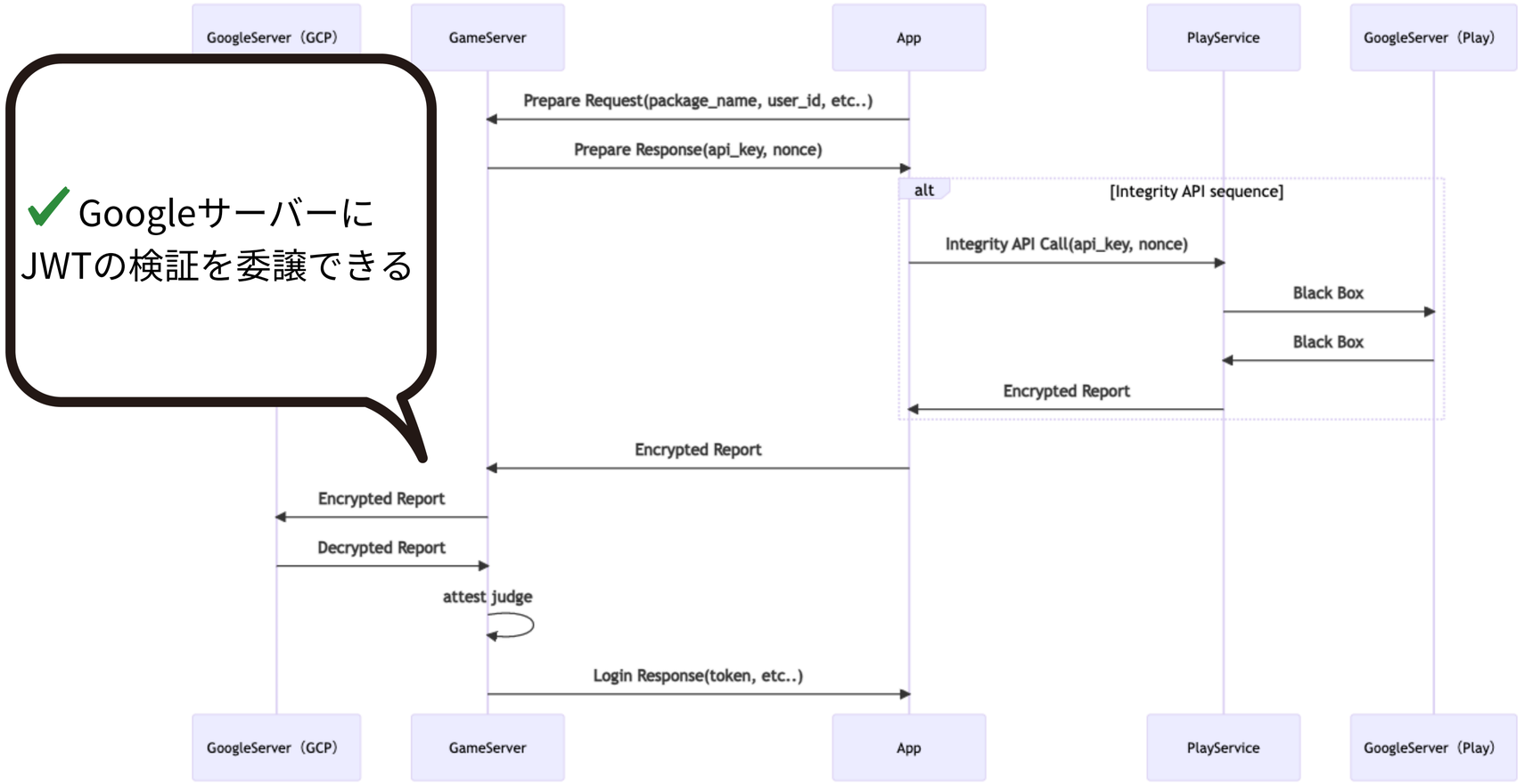
Improvements from "SafetyNet"

Play Integrity APIの処理フロー



Improvements from "SafetyNet"

Play Integrity APIの処理フロー



✔ Googleサーバーに
JWTの検証を委譲できる

Improvements from "SafetyNet"

SafetyNet Attestation APIからPlay Integrity APIでの移行で得られるメリット

- **レポートの検証にGoogleサーバが使用できるようになった**

  SafetyNetでは自前でJWTの検証が必要だった

- **各ドメインレベルのレポートが他のレポートへ干渉することがない**

  SafetyNetでは検証環境が要件に満たない場合，正常系と異なるフォーマットのJSONが返却されていた

- **ライセンス認証ができるようになった**

  有料アプリを正規に購入したユーザーかどうかを検証できるようになった

- **Unity用SDKが配布されるようになった**

  これまではUnityに実装する場合，native SDKを実装する必要があった

SafetyNet Response

```
{
    //Googleサーバーによって算出されるSafetyNet Attestation APIをリクエストした日時.
    "timestampMs": 9860437986543,
    //リプレイ攻撃を防ぐためのnonce.
    "nonce": "R2Rra24fVm5xa2Mg",
    //SafetyNet Attestation APIにてリクエストしたアプリのパッケージ名.
    "apkPackageName": "com.package.name",
    //SafetyNet Attestation APIにてリクエストしたアプリケーションの開発者証明書ハッシュ.
    "apkCertificateDigestSha256": ["base64 encoded, SHA-256 hash of the certificate"],
    //Compatibility Test Suiteに合格した端末か否か.
    "ctsProfileMatch": true,
    //ctsProfileMatchの値を含む，より詳細なデバイスの正当性検証結果.
    "basicIntegrity": true
}
```

SafetyNet Response

```
{
    //Googleサーバーによって算出されるSafetyNet Attestation APIをリクエストした日時.
    "timestampMs": 9860437986543,
    //リプレイ攻撃を防ぐためのn
    "nonce": "R2Rra24fVm
    //SafetyNet Attestation
    "apkPackageName": "co
    //SafetyNet Attestation
    "apkCertificateDiges
    //Compatibility Test Suiteに含
    "ctsProfileMatch": true,
    //ctsProfileMatchの値を含む，よ  詳細なデバイスの正当性検証結果.
    "basicIntegrity": true
}
```

レスポンスのJSONフォーマットはこれだけではない

SafetyNet Unexpected Response

判定対象のデバイスが基準に満たない場合

```
{
    //デバイスを判定基準まで回復する方法が列挙される.
    "advice": ["LOCK_BOOTLOADER, RESTORE_TO_FACTORY_ROM"]
}
```

想定外のエラーが発生した場合

```
{
    //エラーの内容が記載される.
    "error": "Encoded error information relevant to the current API request."
}
```

Integrity API Response

```
{
  //リクエストの詳細セクション
  //過去の正常系判定レスポンスを保持して，チート時に不正判定レポートデータを過去の正常系に上書きするReplay-Attackへの対策として，
      レスポンス内容がリクエストに対応しているかを判定するために使用します.
 "requestDetails": {
  //・・・
 },

  //アプリケーションの完全性セクション
  //アプリケーションの詳細情報を判定するために有効な情報が格納されています.
 "appIntegrity": {
  //・・・
 },

  //デバイスの完全性セクション
  //アプリケーション実行環境の正当性を判定するために有効な情報が格納されています.
 "deviceIntegrity": {
  //・・・
 },

  //アカウントの詳細セクション
  //アプリケーションの利用資格を判定するために有効な情報が格納されています.
 "accountDetails": {
  //・・・
 }
}
```

Integrity API Response

```
{
  "requestDetails": {
    //Integrity APIにてリクエストしたアプリケーションのパッケージ名.
    "requestPackageName": "com.package.name",
    //リプレイ攻撃を防ぐためのnonce.
    "nonce": "aGVsbG8gd29scmQgdGhlcmU",
    //Googleサーバーによって算出されるIntegrity APIをリクエストした日時.
    "timestampMillis": 1617893780
  },
  "appIntegrity": {
    //アプリケーションの検証結果，下記の3つの状態の1つが決定される.
    //PLAY_RECOGNIZED = Playで配信された情報と一致する, UNRECOGNIZED_VERSION = Play Storeの情報と不一致, UNEVALUATED = アプリケーションの実行環境が検証に必要な要件を満たしていない.
    "appRecognitionVerdict": "PLAY_RECOGNIZED",
    //Integrity APIにてリクエストしたアプリケーションのパッケージ名.
    //appRecognitionVerdictがUNEVALUATED以外の場合，このフィールドが記載されます.
    "packageName": "com.package.name",
    //Integrity APIにてリクエストしたアプリケーションの開発者証明書ハッシュ.
    //appRecognitionVerdictがUNEVALUATED以外の場合，このフィールドが記載されます.
    "certificateSha256Digest": ["base64 encoded, SHA-256 hash of the certificate"],
    //Integrity APIにてリクエストしたアプリケーションのバージョンコード.
    //appRecognitionVerdictがUNEVALUATED以外の場合，このフィールドが記載されます.
    "versionCode": 1
  },
  "deviceIntegrity": {
    //デバイスの正当性検証結果.
    "deviceRecognitionVerdict": ["MEETS_DEVICE_INTEGRITY"]
  },
  "accountDetails": {
    //アプリケーションの利用アカウントの検証結果，下記の3つの状態の1つが決定される.
    //LICENSED = 利用資格を持ったユーザーがPlay Storeから入手したアプリケーションを使用しています.
    //UNLICENSED = ユーザーはPlay Store以外から入手したアプリケーションを使用しています.
    //UNEVALUATED = アプリケーションの実行環境が検証に必要な要件を満たしていません.
    "appLicensingVerdict": "LICENSED"
  }
}
```

Improvements from "SafetyNet"

SafetyNet Attestation APIからPlay Integrity APIでの移行で得られるメリット

- **レポートの検証にGoogleサーバが使用できるようになった**

  SafetyNetでは自前でJWTの検証が必要だった

- **各ドメインレベルのレポートが他のレポートへ干渉することがない**

  SafetyNetでは検証環境が要件に満たない場合，正常系と異なるフォーマットのJSONが返却されていた

- **ライセンス認証ができるようになった**

  有料アプリを正規に購入したユーザーかどうかを検証できるようになった

- **Unity用SDKが配布されるようになった**

  これまではUnityに実装する場合，native SDKを実装する必要があった

Integrity API includes a LVL.

Play Integrity APIはLVLの機能を内包している.

- 不正コピー防止のために利用されてきたLVL
  （License Verification Library）の機能がレスポンスに含まれる

LVLの機能では自前でJWTの検証が必要だった
この機能はIntegrity APIレスポンスのaccountDetailsセクション内，appLicensingVerdictフィールドで確認できる
Integrity APIで代用することで実装の手間を減らせて，かつJWT固有のアンチパターンを回避できるので安全

```
"accountDetails": {
  //アプリケーションの利用アカウントの検証結果，下記の3つの状態の1つが決定される.
  //LICENSED, UNLICENSED, UNEVALUATED
  "appLicensingVerdict": "LICENSED"
}
```

Integrity API includes a LVL.

Play Integrity APIでLVLの代用することのデメリット.

- **LVLと比べて実行環境のデバイスの状態で判定ができない場合がある**

  appLicensingVerdictのフィールドには，UNEVALUATEDというアプリケーションの実行環境が
  検証に必要な要件を満たしていない場合の判定結果が含まれるため，LVLと異なり実装方法に工夫が必要

- **LVLと比べてAPIの呼び出し回数に制限がある**

  SafetyNetとPlay Integrity APIにはそれ自体に後述の呼び出し制限があるため，LVLと異なり実装方法に工夫が必要

Improvements from "SafetyNet"

SafetyNet Attestation APIからPlay Integrity APIでの移行で得られるメリット

- **レポートの検証にGoogleサーバが使用できるようになった**

  SafetyNetでは自前でJWTの検証が必要だった

- **各ドメインレベルのレポートが他のレポートへ干渉することがない**

  SafetyNetでは検証環境が要件に満たない場合，正常系と異なるフォーマットのJSONが返却されていた

- **ライセンス認証ができるようになった**

  有料アプリを正規に購入したユーザーかどうかを検証できるようになった

- **Unity用SDKが配布されるようになった**

  これまではUnityに実装する場合，native SDKを実装する必要があった

Get started with the Integrity API

Play Integrity APIを使用するための操作手順

1. **Google Cloud Consoleにてプロジェクトを作成し，Play Integrity APIを有効にする．**
2. **Google Cloud ConsoleにてAPIキーを作成し，[APIの制限]設定メニューからキーの有効範囲をPlay Integrity APIに限定する．**

   セキュアにAPIを叩くため，APIキーが割れる前提で不正利用を防ぐためにAPIキーの使用を限定的に制限してください

3. **Google Play Consoleにて対象のアプリケーションプロジェクトの左ペインにある[設定]メニュー内，[アプリの完全性]を選択する．**

   この設定には最低限のアプリケーション情報をPlayStoreに登録する必要がある．なお，この段階ではアプリケーションバイナリの登録は不要
   初期設定画面に従って手順１で作成したプロジェクトを選択する

4. **UnityプロジェクトにUnityパッケージをインポートする．**

   https://developers.google.com/unity/packages#play_integrity_api からダウンロード可能

5. **API使用量ティアの引き上げ申請を行う**

   API使用量ティア（APIの利用回数制限）は初期状態で10,000リクエスト/日なので必要であれば引き上げ申請を行う
   https://support.google.com/googleplay/android-developer/contact/piaqr にて申請を行うことができる．申請は無料で行えるが審査で妥当な理由が必要となる

以上で最小限の利用準備が完了！

Effective Integrity API

Play Integrity APIを利用する上で重要な事項

- **レポートの検証はゲームサーバーとGoogleサーバで連携して行う**

  改竄検知の改竄を防ぐために，必ずIntegrity APIのレスポンスはクライアントでパースせず，
  ゲームサーバーとGoogleサーバー上でやりとりして，判定もゲームサーバー上で処理する

- **通信の回数が多いので異常系を常に考慮すること**

  ログイン処理時に判定を行うことでAPI呼び出しを最小回数にできるが，Googleサーバーがダウンしていたり，
  使用量ティアによるエラーを考慮して，判定結果をもとに即時ログインブロックなどはしない

- **PlayStoreを利用しないAndroidベースのOSでは使用できないことを考慮すること**

  KindleやWindows11，Googleとの通信が制限されているなどの理由でPlayが入っていない端末では判定できない

Other impressions

Play Integrity APIのUnity package所感

- **展開してみるとnativeライブラリをラップしてインタフェースを生やしているようだが・・**

  Proguard（難読化）設定に多くの箇所が含まれていないため，耐タンパ性に欠ける

```
-keep class com.google.android.play.core.integrity.IntegrityManager {
  public abstract com.google.android.play.core.tasks.Task requestIntegrityToken(com.google.android.play.core.integrity.IntegrityTokenRequest);
}

-keep class com.google.android.play.core.integrity.IntegrityManagerFactory {
  public static com.google.android.play.core.integrity.IntegrityManager create(android.content.Context);
}

-keep class com.google.android.play.core.integrity.IntegrityTokenRequest {
  public abstract java.lang.String nonce();
  public abstract java.lang.Long cloudProjectNumber();
  public static com.google.android.play.core.integrity.IntegrityTokenRequest$Builder builder();
}

-keep class com.google.android.play.core.integrity.IntegrityTokenRequest$Builder {
  public abstract com.google.android.play.core.integrity.IntegrityTokenRequest$Builder setNonce(java.lang.String);
  public abstract com.google.android.play.core.integrity.IntegrityTokenRequest$Builder setCloudProjectNumber(long);
  public abstract com.google.android.play.core.integrity.IntegrityTokenRequest build();
}

-keep class com.google.android.play.core.integrity.IntegrityTokenResponse {
  public abstract java.lang.String token();
}

-keep class com.google.android.play.core.integrity.model.IntegrityErrorCode {
    public static int NO_ERROR;
    public static int API_NOT_AVAILABLE;
    public static int PLAY_STORE_NOT_FOUND;
```

Thanks!