

# G-PRESTO 사용 가이드

SDK 연동 기본 가이드 2.8.1

## 목차

G-Presto 쉽게 적용하기 .....	3
회원가입 및 로그인 .....	4
G-Presto 추가정보 입력 .....	5
패키지 설정(게임 정보 등록) .....	6
권한 설정 .....	7
사이트 SDK 적용 메뉴 설명 .....	8
SIGNING .....	9
자동 서명 기능 절차 안내 .....	10
수동 SIGNING .....	13
UUID 를 통한 치트유저 확인 .....	15
UNITY 플러그인 프로젝트 설정 .....	16
UNITY 플러그인 운영체제별 적용 .....	17
G-Presto AOS UNITY 실행 코드 적용 .....	18
UNITY 플러그인 변수 암호화 사용 예제 .....	18
UNITY 플러그인 스피드핵 감지 사용 예제 .....	20
UNITY 메타 데이터 암호화 .....	21
G-PRESTO 난독화 정상 적용 테스트 .....	23
G-PRESTO 치트앱 차단 테스트 .....	24
사용자 정보 확인 .....	25
G-Presto 가 제공하는 API (getEmul) .....	26
G-PRESTO Log Data 참고사항 .....	27
G-PRESTO Log Data 요청 .....	27
G-PRESTO Blacklist 요청 .....	28
G-PRESTO 해킹커뮤니티 배포 앱 사용자 Data 요청 .....	29
빌드툴 통합 .....	32
Android Store app signing 관련 변경 사항 안내 .....	34
Android app bundle .....	36
Android 11 이상의 Query_ALL_packages 권한 요구 .....	38
자주 묻는 질문 FAQ .....	39

## G-Presto 쉽게 적용하기

### G-PRESTO 쉽게 적용하기

※ 간략하게 적용할 수 있는 절차만 안내되어 있습니다. 상세한 내용은 가이드를 참고 부탁드립니다.

1. console.largosoft.co.kr 에 가입합니다.

#### ■ 회원가입 및 로그인

2. 패키지 설정 메뉴에서 보안 적용할 패키지 및 게임명을 등록합니다.

#### ■ 패키지 설정(게임 정보 등록)

3. 등록된 패키지를 G-Presto 담당자에게 권한을 요청합니다.

4. 게임 빌드에 G-Presto UUID 매칭 작업을 진행합니다. (필수 작업)

#### ■ 권한 설정

#### ■ UUID 를 통한 치트유저 확인

5. apk, aab 에 사용할 서명 정보를 등록하여 보안적용과 서명을 같이 진행할 수 있습니다.

#### ■ 자동 서명 기능 절차 안내

6. 콘솔 페이지 G-Presto 보안 적용 메뉴에서 보안을 적용합니다.

#### ■ 사이트 SDK 적용 메뉴 설명

7. Signed\_xxx.apk 혹은 UnSigned\_xxx.apk 등 보안 적용된 앱을 실행하여 정상적으로 동작하는지 테스트합니다.

#### ■ G-PRESTO 난독화 정상 적용 테스트

#### ■ G-PRESTO 치트앱 차단 테스트

8. Unity 일 경우 Unity 플러그인 적용 및 Unity Meta-data 암호화 적용으로 보안을 강화할 수 있습니다.

#### ■ UNITY 플러그인 프로젝트 설정

#### ■ UNITY 메타 데이터 암호화

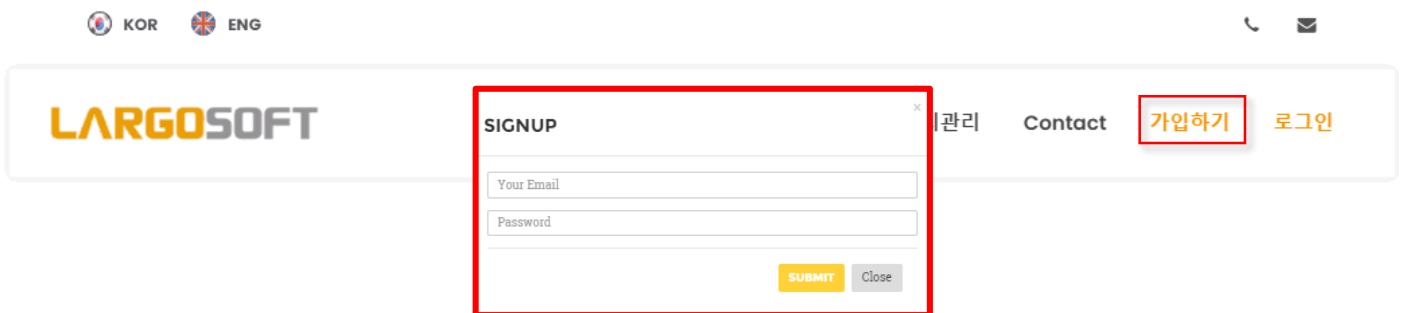
9. G-Presto 관리자에게 전달받은 G-Presto 서버와 게임 서버 통신 기반인 G-Presto 크로스체크를 적용하여 로그를 기록하게 처리하고, 충분한 테스트를 진행하여 보안을 강화할 수 있습니다.

## G-PRESTO SDK 적용 사전작업

### 회원가입 및 로그인

- LARGOSOFT 콘솔 사이트 – <https://console.largosoft.co.kr>

1. LARGOSOFT 사이트 방문 후 가입하기
2. 인증메일 확인
3. LARGOSOFT 사이트 로그인 > G-Presto 사이트로 자동이동
4. 향후 G-Presto 사이트로 직접 로그인



### 모바일게임보안솔루션 G-Presto는

해커의 해킹 시도율은 낮추고 일반 유저로의 전환율을 높입니다.



#### Largosoft G-Presto Authentication Mail

보낸사람: ☆ <help@largosoft.co.kr>  
받는사람

안녕하세요, 라르고소프트 G-Presto 가입을 진심으로 환영 합니다.

아래의 링크를 클릭하시면 자동으로 인증이 완료 됩니다.

[http://largosoft.co.kr/EmailValidation?user= &auth=e6d0f359a8c0add5c51e872c6258cf5df53cbfbd](http://largosoft.co.kr/EmailValidation?user=&auth=e6d0f359a8c0add5c51e872c6258cf5df53cbfbd)



## G-PRESTO SDK 적용 사전작업

### 패키지 설정(게임 정보 등록)

패키지 설정

Home > 패키지 설정

패키지 등록

NOTE: 패키지 등록 완료 이후 반드시 라르고소프트에 인증 요청 및 승인을 거쳐야 정상적으로 이용 가능 합니다.

게임명 등록

게임명

게임명 등록

패키지명 등록

OS종류 OS종류를 선택해 주시기 바랍니다.
패키지명
게임명 게임명을 선택해 주시기 바랍니다.
패키지명 등록

패키지 설정 옵션 설명

패키지 설정

게임명

Search:

OS종류	구분	만료일	게임명	패키지명	치트돌차단	앱위변조차단	가상머신차단	매크로차단	dex난독화
Android	Unauthorized		crashtest	com.example4563.simpleplayer	OFF	OFF	OFF	OFF	OFF
Android	Authorized	2020-04-30	crashtest	com.largounity.test	OFF	OFF	OFF	OFF	ON

1. 게임 명 등록 – 개발사에서 향후 통계분석 툴 확인 시 프로젝트를 구분 짓기 위한 분류명.
2. 패키지 명 등록 – 게임명과 매칭되어 해킹통계구분에 사용됨.

#### 해당 내용까지 완료 시 사전작업 완료

3. 하단의 패키지 설정을 통해 게임 별 보안 설정을 진행함.
4. 패키지 설정에 어려움이 있어 예시가 필요 시 패키지 설정 옵션 설명을 참고

## G-PRESTO SDK 적용

### 권한 설정

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bishopsoft.Presto"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="17" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.bishopsoft.Presto.MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```


<AndroidManifest.xml>

AndroidManifest.xml 내부에 상기 2 개의 권한이 입력되어 있어야 정상적으로 Presto 솔루션이 적용됨.

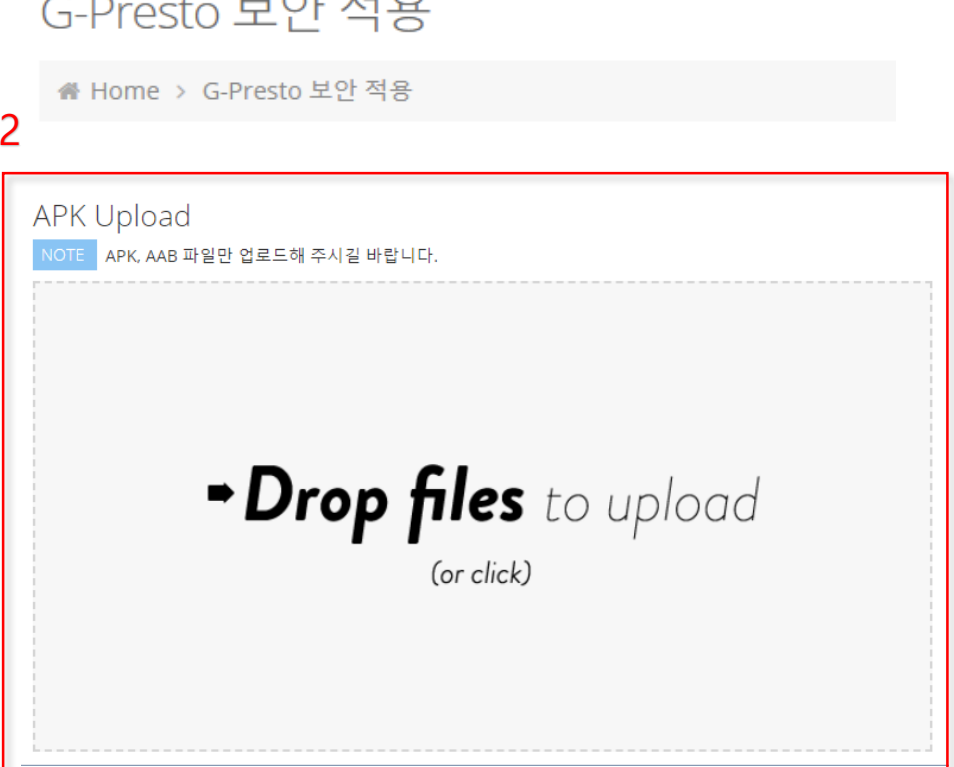
## G-PRESTO SDK 적용

### 사이트 SDK 적용 메뉴 설명

**1**



**2**



1. G-Presto 보안 적용 메뉴.
2. 난독화를 진행할 APK 파일을 **업로드** 함.

이하의 작업은 1 번메뉴를 누른 후 진행.



## G-PRESTO SDK 적용

### ALIAS 및 KEYSTORE

개발사에서 마켓등록을 위해 개발 시 생성하는 파일 및 정보.

해당파일을 SDK 적용시 업로드 하거나 수동으로 Signing 작업을 진행 해야 함.

키 생성 및 사인 : <http://www.androidpub.com/4742>

Signing 관련 구글 개발자 사이트 : <http://developer.android.com/tools/publishing/app-signing.html>

### SIGNING

G-PRESTO SDK 적용 후 마켓등록을 위하여 개발사의 App Signing 을 거쳐야 함.

자동 Signing : 개발사에서 생성한 Keystore 파일 등록 후 이후에는 APK 만 업로드 하여도 난독화 및 SDK 적용후 Signing 까지 자동 적용되어 파일 다운로드 됨. 저장된 keystore 파일 변경 및 삭제가 필요한 경우 라르고소프트에 문의

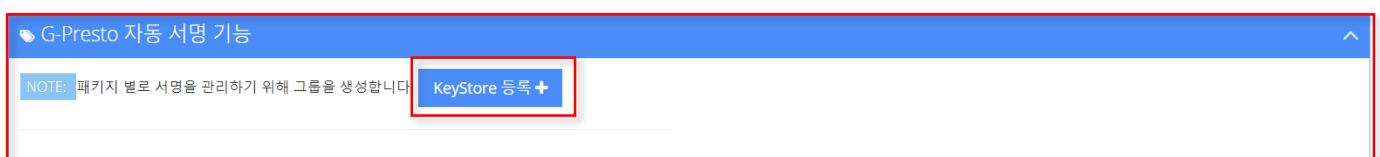
수동 Signing : keystore 파일 업로드가 어려울 경우 APK 에 난독화 및 SDK 적용 후 다운로드 된 APK 파일에 수동 Signing 작업을 진행.

### 자동 서명 기능

프로젝트의 패키지, 혹은 앱 마켓 별로 서명의 정보가 다를 수 있기 때문에 프로젝트 단위 별로 서명 정보를 관리 할 수 있음.

1. G-Presto 자동 서명 기능란의 KeyStore 등록 버튼을 누름.
2. 절차에 따라 프로젝트명, KeyStore 정보 입력, KeyStore 파일 업로드 후 등록
3. 자동 서명 기능에 프로젝트 명으로 생성된 박스를 클릭하여 등록된 서명정보와 매칭될 패키지 설정
4. APK 를 업로드하여 보안 적용
5. 적용 완료 후 자동으로 완료된 파일이 다운로드 됨

\* 자동 Signing 결과물 : signed\_파일명



## G-PRESTO SDK 적용

### 자동 서명 기능 절차 안내

프로젝트 등록 - Step 1 of 4

1 프로젝트 등록 2 KeyStore 정보 입력 3 KeyStore 파일 업로드 4 입력 정보 확인

프로젝트명 설정

프로젝트명 \*

자동 서명 기능을 이용할 패키지와 KeyStore 간의 관계 설정을 위한 프로젝트명입니다.

다음 >

2. 등록할 서명 정보를 관리하기 위한 프로젝트명 입력

프로젝트 등록 - Step 2 of 4

1 프로젝트 등록 2 KeyStore 정보 입력 3 KeyStore 파일 업로드 4 입력 정보 확인

Keystore 정보 입력

AliasName \*

등록할 Keystore 정보의 AliasName을 입력해주세요.

Keystore 패스워드 \*

등록할 Keystore 정보의 Keystore 패스워드를 입력해주세요.

Key 패스워드

키 생성 시 별도로 Key 패스워드를 Keystore 패스워드와 다르게 입력한 경우만 입력하세요.  
기본적으로는 공란으로 두시기 바랍니다.  
공란인 경우 자동으로 storepass와 동일하게 저장됩니다.

뒤로가기 < 다음 >

1. KeyStore 정보를 입력

프로젝트 등록 - Step 3 of 4

1 프로젝트 등록 2 KeyStore 정보 입력 3 KeyStore 파일 업로드 4 입력 정보 확인

Keystore 파일 업로드

Keystore 파일을 업로드해주세요. \*

Drop files to upload  
(or click)

keyStore는 최초 1회만 업로드하면 다시 업로드하실 필요 없습니다.  
.jks, .keyStore 확장자만 지원합니다.

뒤로가기 < 다음 >

5. 등록할 KeyStore 파일을 업로드

업로드 실패 시 문의 바랍니다.

업로드 성공

파일 업로드에 성공했습니다.

OK

프로젝트 등록 - Step 4 of 4

1 프로젝트 등록 2 KeyStore 정보 입력 3 KeyStore 파일 업로드 4 입력 정보 확인

입력한 정보를 확인합니다.

프로젝트 : 관리할 프로젝트명

Keystore

Alias Name : Keystore AliasName

Keystore Password : KeystorePassword

Key Password : KeyPassword

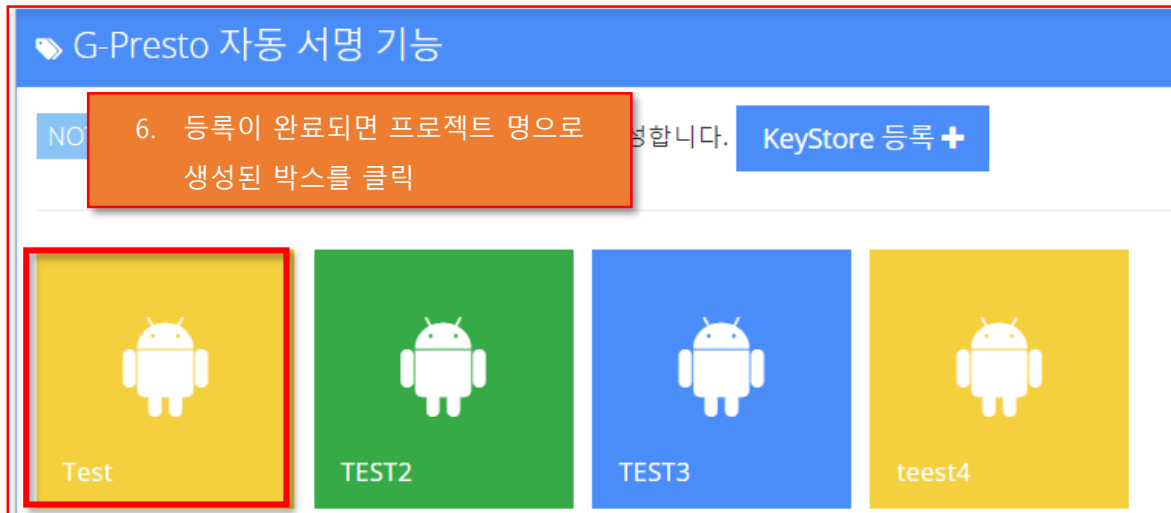
Uploaded File : keystore.jks

뒤로가기 < 등록 >

3. 입력한 정보를 확인

4. 이상이 없다면 등록 버튼 클릭

## G-PRESTO SDK 적용



등록한 서명 정보를 수정해야하는 경우  
정보수정 버튼을 이용하여 수정이 가능

주의) KeyStore 파일을 변경해야 할 경우  
문의하시기 바랍니다.

7. 등록된 패키지목록에서 등록된 서명  
정보로 서명할 패키지를 ON/OFF  
옵션 변경으로 적용가능

8. 저장 버튼을 눌러 패키지  
설정 사항을 적용

## G-PRESTO SDK 적용

### APK Upload

**NOTE** APK, AAB 파일만 업로드해 주시길 바랍니다.

→ **Drop files** to upload  
(or click)

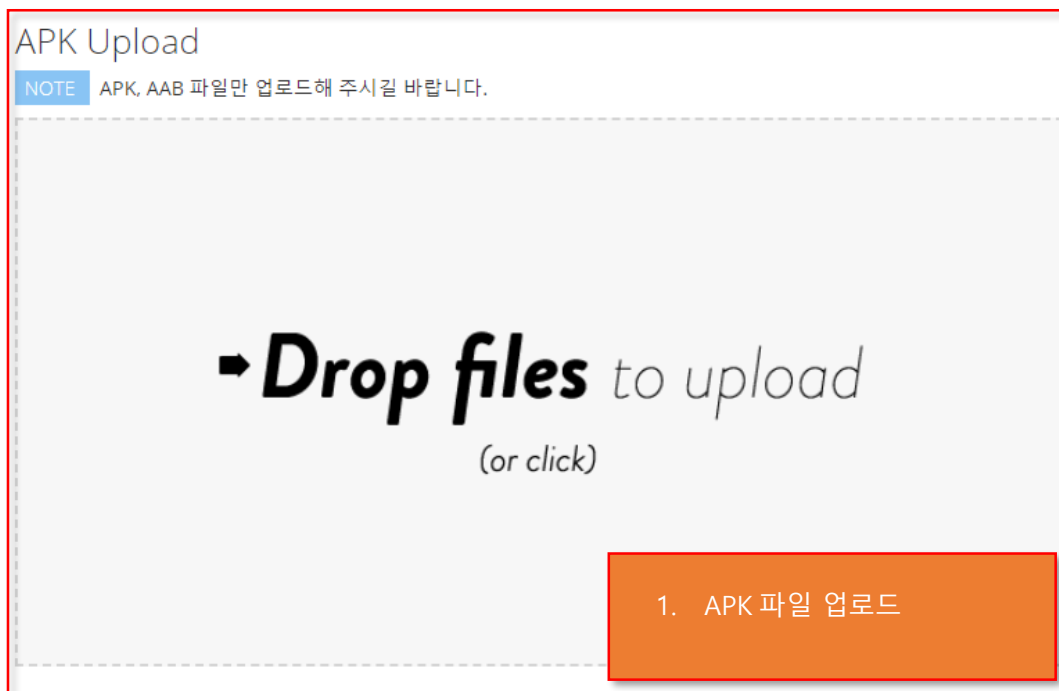
9. APK 파일 업로드

## G-PRESTO SDK 적용

### 수동 SIGNING

1. 앱(APK) 파일을 드래그 앤 드랍 또는 Upload 을 찾아서 업로드(업로드 이후 1-3 분 정도 작업 소요됨)
2. 적용 완료 후 자동으로 완료된 파일이 다운로드 됨.

\* 수동 Signing 결과물 : Unsigned\_파일명.apk, Unsigned\_파일명.aab



3. 이미 Keystore 가 생성된 상태라고 가정하고 keystore 생성하는 방법은 생략.
4. AAB 인 경우 jarsigner, APK 인 경우 Apksigner 를 이용하여 서명을 진행함.

참고 URL

Apksigner 로 서명하기 : <https://developer.android.com/studio/command-line/apksigner?hl=ko>

Android 개발자 문서 : <http://developer.android.com/tools/publishing/app-signing.html>

Jarsigner 로 AAB 서명하기

```
jarsigner -verbose -sigalg SHA256withRSA -digestalg SHA-256 -keystore "key_path" -storepass "store_password" -keypass "key_password" "myAAB.aab" "key_alias"
```

\*AAB 파일은 서명 이후 zipalign 따로 하지 않아도 됩니다.

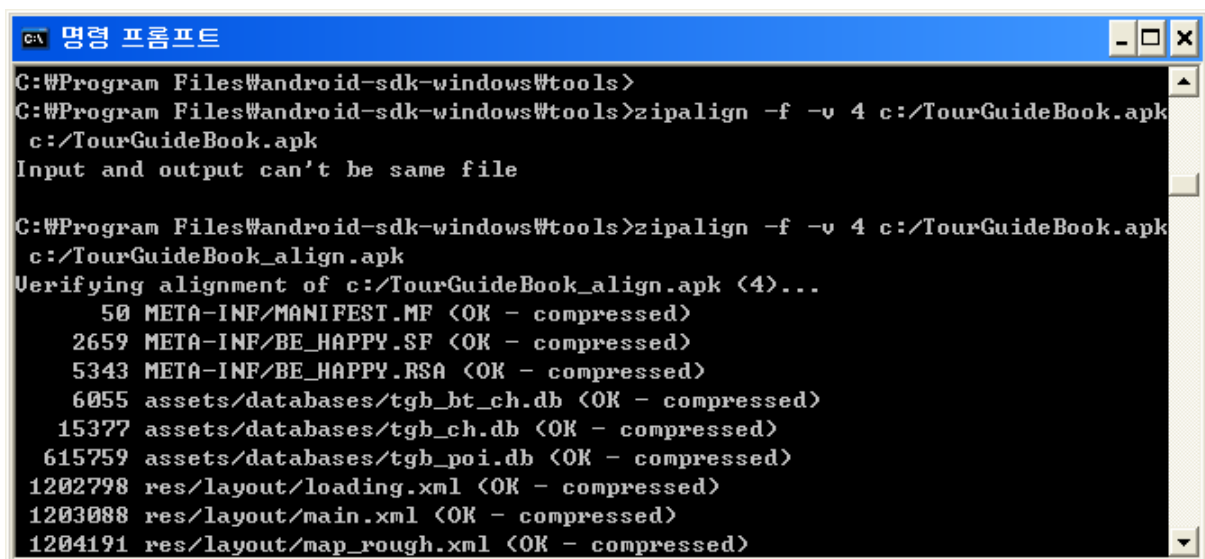
## G-PRESTO SDK 적용

### 수동 Zipalign

1. Unsigned\_파일명.apk 를 다운로드 이후 Zipalign 를 반드시 진행.  
 유효한 서명 값을 가지기 위해 Zipalign 작업을 진행 후 서명 작업을 진행.  
 Zipalign 시 파일 접근 시 읽기 성능이 향상 된다고 알려짐

```
zipalign [-f] [-v] <alignment> infile.apk outfile.apk
zipalign -f -v 4 c:/TourGuideBook.apk c:/TourGuideBook.apk
```

- f: 파일이 존재 하면 덮어쓰기
- v: 자세한 정보 출력 (zipalign 되는 과정을 출력한다.)
- c: 주어진 파일 정렬 확인



```
C:\Program Files\android-sdk-windows\tools>
C:\Program Files\android-sdk-windows\tools>zipalign -f -v 4 c:/TourGuideBook.apk
c:/TourGuideBook.apk
Input and output can't be same file

C:\Program Files\android-sdk-windows\tools>zipalign -f -v 4 c:/TourGuideBook.apk
c:/TourGuideBook_align.apk
Verifying alignment of c:/TourGuideBook_align.apk (4)...
 50 META-INF/MANIFEST.MF (OK - compressed)
2659 META-INF/BE_HAPPY.SF (OK - compressed)
5343 META-INF/BE_HAPPY.RSA (OK - compressed)
6055 assets/databases/tgb_bt_ch.db (OK - compressed)
15377 assets/databases/tgb_ch.db (OK - compressed)
615759 assets/databases/tgb_poi.db (OK - compressed)
1202798 res/layout/loading.xml (OK - compressed)
1203088 res/layout/main.xml (OK - compressed)
1204191 res/layout/map_rough.xml (OK - compressed)
```

2. Zipalign 이후 apksigner 를 이용한 서명 진행

```
java -jar apksigner.jar sign -verbose --ks "my-release-key.keystore" --ks-key-alias "alias_name"
--key-pass pass:"key_password" --ks-pass pass:"store_password" --v1-signing-enabled true
--v2-signing-enabled true my_application.apk
```

## G-PRESTO UUID 생성

### UUID 를 통한 치트유저 확인

G-Presto UUID 와 게임 서버의 UUID 를 매칭하여 게임 서버에서의 유저 추적 및 차단을 위해 반드시 적용해야 합니다.

G-Presto UUID 는 Android ID 와 FINGERPRINT 값을 조합하여 생성하기 때문에 기기를 초기화하거나, 앱을 재생성할 경우 변경될 수 있습니다. 또한 에뮬레이터 복제와 같은 특수 상황에서 중복 UUID 가 발생할 수 있습니다.

G-Presto UUID 와 매칭되는 게임 아이디를 기반으로 해킹 유저를 차단하실 수 있습니다.

<Unity 용 샘플 코드>

```
//G-Presto UUID(Android)
//추후 G-Presto 서버와 매칭을 위해 유저의 대한 정보를 게임 서버로 전송시 해당 함수를 호출하여 값을 게임
//서버로 전송
string uuid = GPresto.Common.GPUt i l s . GPUUID ();
Debug.Log ("G-Presto UUID = " + uuid);
```

<Unity 가 아닐 경우 샘플 코드>

```
String androidId, strUUID;
androidId = android.provider.Settings.Secure.getString(getContentResolver(),
android.provider.Settings.Secure.ANDROID_ID);

strUUID = androidId + "_" + Build.FINGERPRINT;
strUUID = getMD5Hash(strUUID);
```

<디바이스 정보 등을 조합하여 UUID 생성>

```
public static String getMD5Hash(String s) {
    MessageDigest m = null;
    String hash = null;
    try {
        m = MessageDigest.getInstance("MD5");
        m.update(s.getBytes(),0,s.length());
        hash = new BigInteger(1, m.digest()).toString(16);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return hash;
}
```

<문자열을 md5 로 변환하는 함수>

## G-PRESTO Unity 플러그인 적용

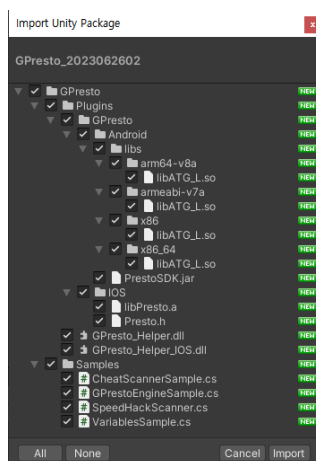
### UNITY 지원환경

Unity 3.x 대부터 지원합니다.

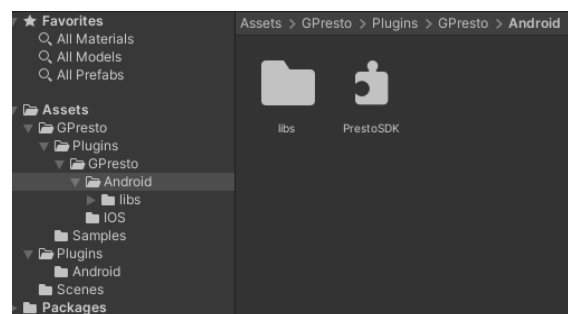
### UNITY 플러그인

G-Presto 는 메모리 치트툴을 통한 변수의 메모리 주소를 찾는 행위를 방지하기 위해 Unity 용 변수 암호화 플러그인을 제공합니다.

### UNITY 플러그인 프로젝트 설정



<Unity 플러그인 import 전>



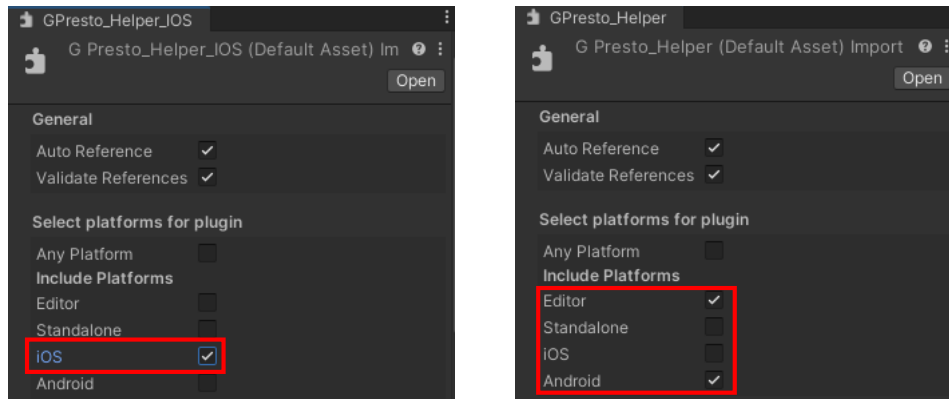
<Unity 플러그인 import 후>

1. 전달 받은 GPresto\_[버전정보].unitypackage 파일을 실행
2. 적용시킬 Unity 프로젝트를 선택
3. <부분1> import를 클릭 시 플러그인 설정 완료



## G-PRESTO Unity 플러그인 적용

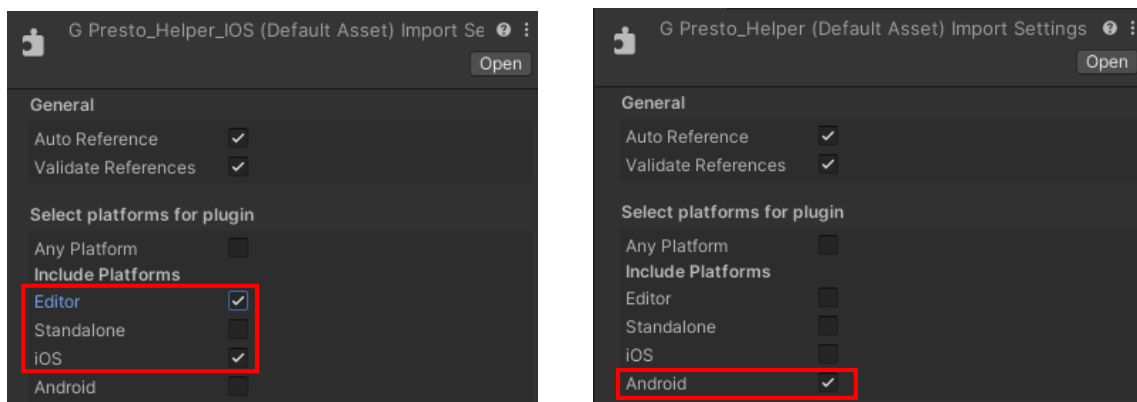
### UNITY 플러그인 운영체제별 적용



<Android 빌드의 경우>

GPresto\_Helper.dll – Android, Editor 선택

GPresto\_Helper\_IOS.dll – iOS 만 선택, Editor 는 선택하지 않음.



<iOS 빌드의 경우>

GPresto\_Helper\_IOS.dll – iOS, Editor 선택

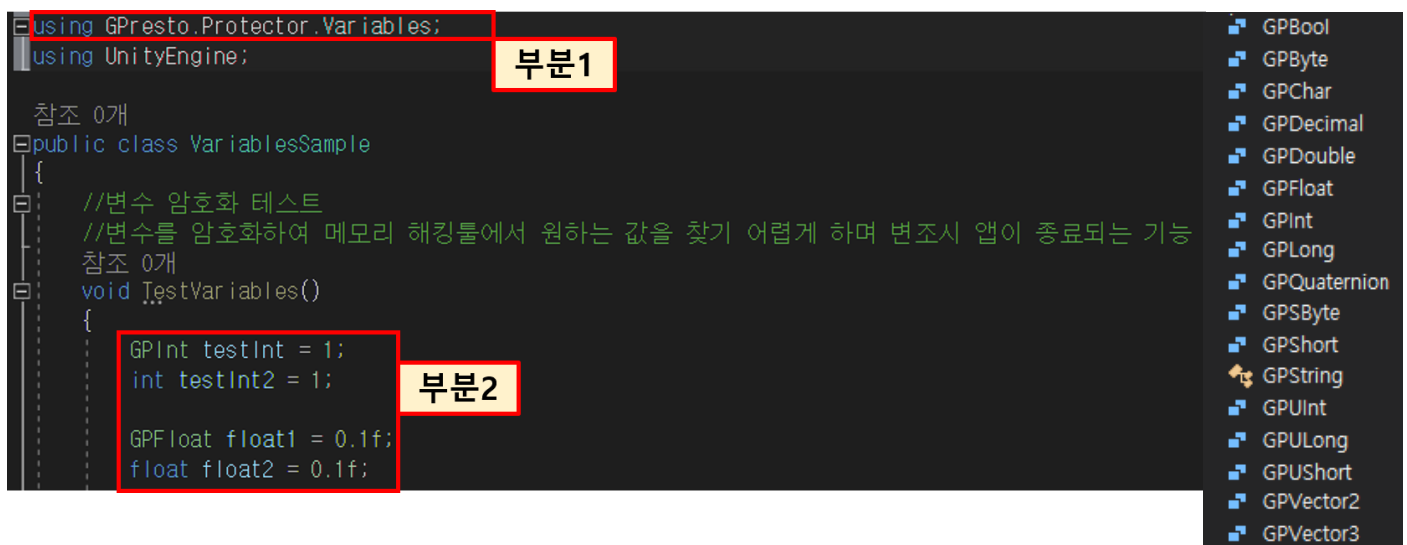
GPresto\_Helper.dll – Android 만 선택, Editor 는 선택하지 않음.

## G-PRESTO Unity 플러그인 적용

### G-PRESTO AOS UNITY 실행 코드 적용

```
// 앱이 시작되는 지점에서 해당 함수를 호출
GPresto.Protector.Engine.GPrestoEngine.Start();
```

### UNITY 플러그인 변수 암호화 사용 예제



<Unity C#Script 에서 GP 자료형 선언과 사용가능한 GP 자료형 종류>

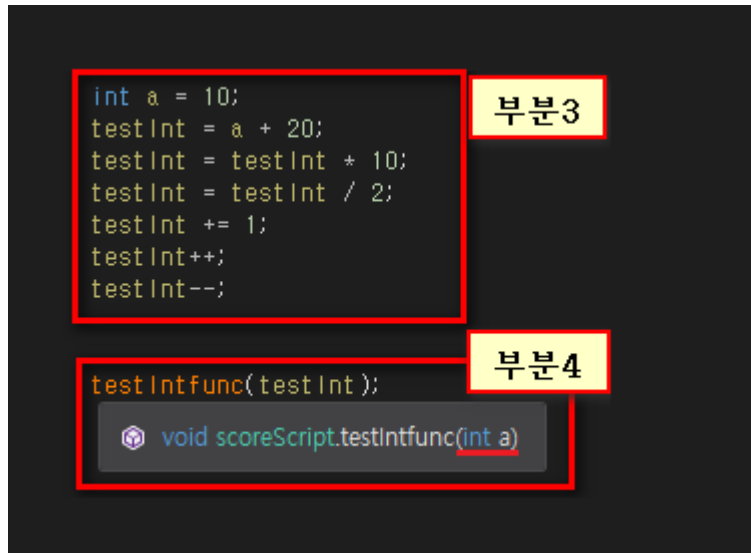
1. <부분1>

G-Presto 자료형을 사용하기 위하여 네임스페이스 추가

2. <부분2>

G-Presto 자료형 선언 예시(일반적인 자료형 선언과 동일하게 사용 가능)

## G-PRESTO Unity 플러그인 적용



< Unity C#Script GPInt 사용 예>

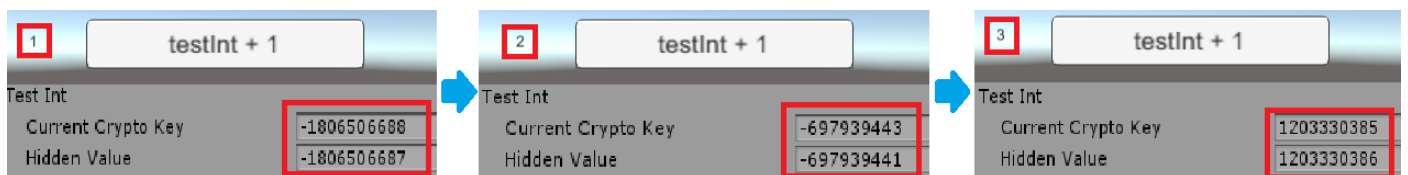
### 3. <부분3>

<부분2>에 선언된 GPInt 자료형 사용 예시(GPInt는 int와 같이 연산이 가능함.)

### 4. <부분4>

<부분2>에 선언된 GPInt 자료형 사용 예시(int형을 매개변수로 하는 testIntfunc함수를 GPInt형인 testInt변수를 전달하여 호출가능 함.)

**\* int 뿐만 아니라 모든 GP자료형들은 기본 자료형과 같은 방식으로 사용 가능함.**

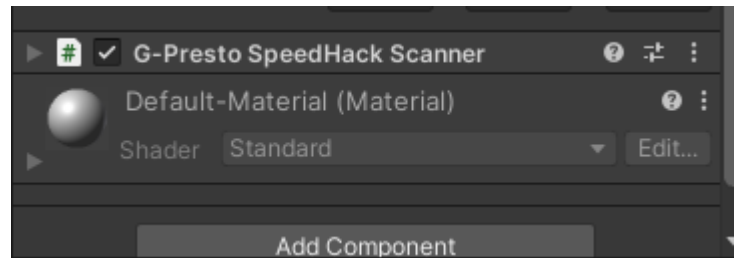


<GPInt testInt 의 암호화된 값>

## G-PRESTO Unity 플러그인 적용

### UNITY 플러그인 스피드해킹 감지 사용 예제

알려지지 않은 스피드해킹을 이용하거나, 가상 컨테이너 환경에서 사용하는 스피드해킹을 감지하여 차단합니다.



<게임 오브젝트에 SpeedHackScanner.cs 를 Add Component - Script>

스피드해킹 차단 기능을 시작하고자 하는 게임 오브젝트에 SpeedHackScanner.cs 를 Add Component - Script.

```
private void Update()
{
    if (!isRunning)
        return;

    if (SpeedHack.IsDetect())
    {
        isRunning = false;

        Debug.LogWarning("SpeedHack Detected");
    }
}
```

< SpeedHackScanner.cs 의 코드 Update() 함수 설명>

SpeedHack 클래스의 isDetect() 함수를 호출하여 스피드해킹을 감지 할 수 있습니다.

G-Presto SDK 에서 제공하는 차단 화면을 출력하고 앱 종료됩니다.

**Unity 플러그인 기능의 차단 기능은 G-Presto 보안 적용이 요구됩니다.**

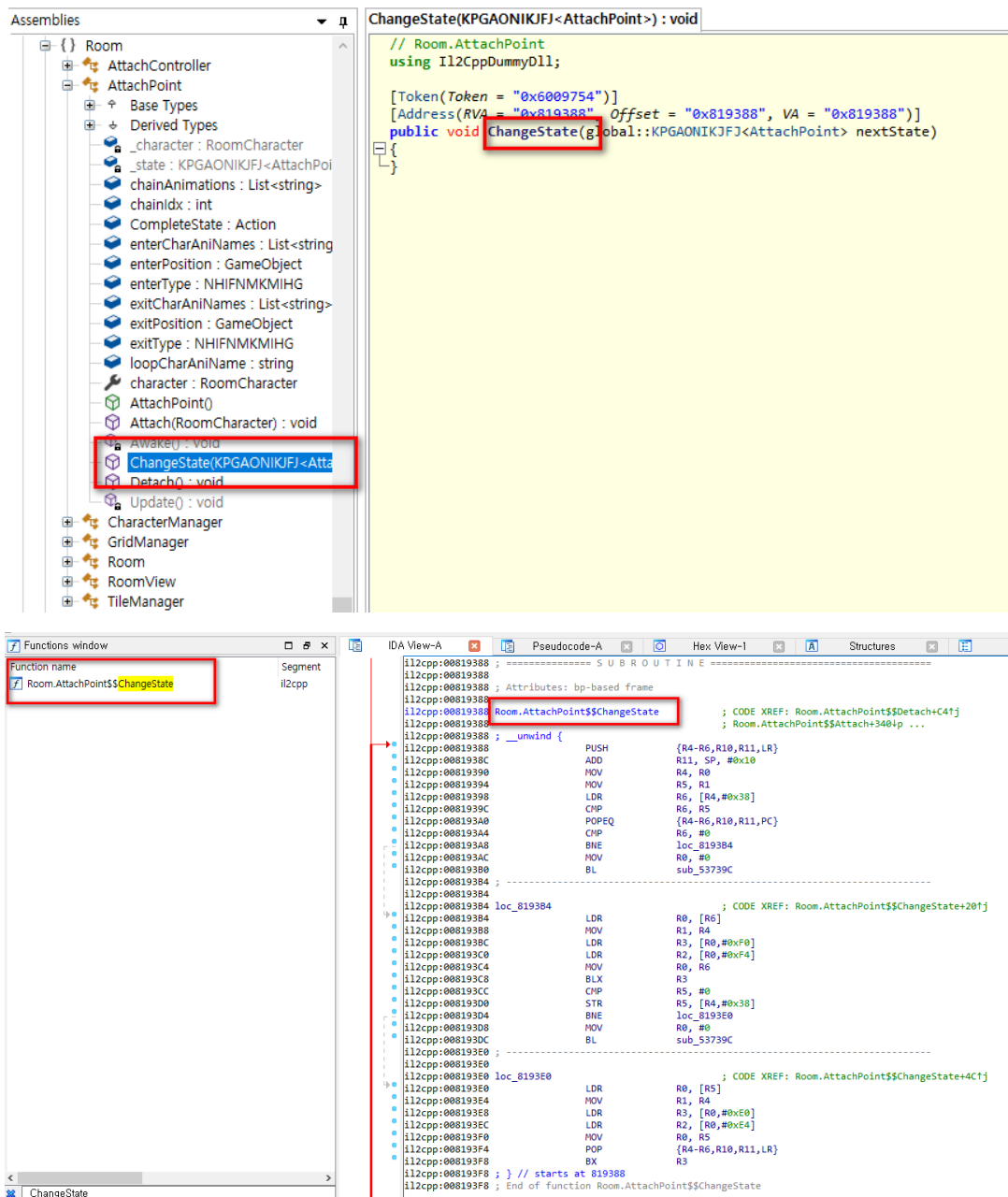
**홈페이지에서 보안 적용 이후 정상적으로 차단 메시지가 확인됩니다.**

**보안 적용이 되지 않은 경우에는 스피드해킹, 메모리변조가 감지되면 앱이 비정상 종료될 수 있습니다.**

# UNITY 메타 데이터 암호화

## UNITY 메타 데이터 암호화

IL2CPP 빌드시 libil2cpp.so 파일에서의 함수 주소 등의 매칭 정보를 저장한 global-metadata.dat 를 암호화하여 앱 실행시 복호화하여 정적 분석을 차단합니다.



The image shows two screenshots. The top screenshot is from the Unity IDE, displaying the 'Assemblies' panel on the left with 'Room' selected, and the 'ChangeState(KPGAONIKJFJ<AttachPoint>) : void' function definition on the right. The bottom screenshot is from IDA Pro, showing the 'Functions window' on the left with 'Room.AttachPoint\$\$ChangeState' selected, and the 'Pseudocode-A' view on the right showing the assembly code for the function.

**Unity Assemblies Panel:**

- Room
  - AttachController
  - AttachPoint
    - Base Types
    - Derived Types
    - \_character : RoomCharacter
    - \_state : KPGAONIKJFJ<AttachPoint>
    - chainAnimations : List<string>
    - chainIdx : int
    - CompleteState : Action
    - enterCharAniNames : List<string>
    - enterPosition : GameObject
    - enterType : NHIFNMKMIHG
    - exitCharAniNames : List<string>
    - exitPosition : GameObject
    - exitType : NHIFNMKMIHG
    - loopCharAniName : string
    - character : RoomCharacter
    - AttachPoint()
      - Attach(RoomCharacter) : void
      - Awake() : void
      - ChangeState(KPGAONIKJFJ<AttachPoint>) : void**
      - Detach() : void
      - Update() : void
  - CharacterManager
  - GridManager
  - Room
  - RoomView
  - TileManager

**Unity ChangeState(KPGAONIKJFJ<AttachPoint>) : void**

```
// Room.AttachPoint
using Il2CppDummyDll;

[Token(Token = "0x6009754")]
[Address(RVA = "0x819388", Offset = "0x819388", VA = "0x819388")]
public void ChangeState(global::KPGAONIKJFJ<AttachPoint> nextState)
{
}
```

**IDA Pro Functions window:**

- Function name: Room.AttachPoint\$\$ChangeState
- Segment: il2cpp

**IDA Pro Pseudocode-A:**

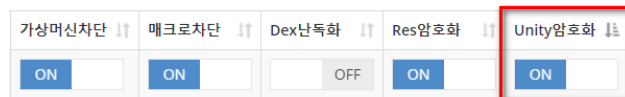
```
112cpp:00819388 ; SUBROUTINE
112cpp:00819388 ; Attributes: bp-based frame
112cpp:00819388 ; Code XREF: Room.AttachPoint$$Detach+C4fj
112cpp:00819388 ; Room.AttachPoint$$Attach+3404p ...
112cpp:00819388 ; _unwind {
112cpp:00819388 PUSH {R4-R6,R10,R11,LR}
112cpp:0081938C ADD R11, SP, #0x10
112cpp:00819390 MOV R4, R0
112cpp:00819394 MOV R5, R1
112cpp:00819398 LDR R6, [R4, #0x38]
112cpp:0081939C CMP R6, R5
112cpp:008193A0 POPEQ {R4-R6,R10,R11,PC}
112cpp:008193A4 CMP R6, #0
112cpp:008193A8 BNE loc_8193B4
112cpp:008193AC MOV R0, #0
112cpp:008193B0 BL sub_53739C
112cpp:008193B4 ; Code XREF: Room.AttachPoint$$ChangeState+20fj
112cpp:008193B4 loc_8193B4 LDR R0, [R6]
112cpp:008193B8 MOV R1, R4
112cpp:008193BC LDR R3, [R0, #0xF0]
112cpp:008193C0 LDR R2, [R0, #0xF4]
112cpp:008193C4 MOV R0, R6
112cpp:008193C8 BLX R3
112cpp:008193CC CMP R5, #0
112cpp:008193D0 STR R5, [R4, #0x38]
112cpp:008193D4 BNE loc_8193E0
112cpp:008193D8 MOV R0, #0
112cpp:008193DC BL sub_53739C
112cpp:008193E0 ; Code XREF: Room.AttachPoint$$ChangeState+4Cfj
112cpp:008193E0 loc_8193E0 LDR R0, [R5]
112cpp:008193E4 MOV R1, R4
112cpp:008193E8 LDR R3, [R0, #0xE0]
112cpp:008193EC LDR R2, [R0, #0xE4]
112cpp:008193F0 MOV R0, R5
112cpp:008193F4 POP {R4-R6,R10,R11,LR}
112cpp:008193F8 BX R3
112cpp:008193F8 ; } // starts at 819388
112cpp:008193F8 ; End of function Room.AttachPoint$$ChangeState
```

<global-metadata.dat 를 통해 매핑된 주소로 함수를 찾아 분석하는 화면>

libil2cpp.so 를 IDA 로 분석하면 원하는 함수 주소를 찾고 분석하기가 매우 어렵습니다만 global-metadata.dat 매칭을 통해 함수명과 주소를 매칭하게 되면 원하는 함수를 검색하여 실제 코드를 분석할 수 있습니다. 따라서 global-metadata.dat 를 암호화하여 분석을 어렵게 만들 수 있습니다. 보안 향상을 위해 사용을 권장합니다.

## UNITY 메타 데이터 암호화

1. 사용 유니티 버전을 확인 후 적용 사이트의 '[Unity MetaData 암호화 사전작업 다운로드](#)' 메뉴에서 사용 유니티 버전과 동일한 파일을 다운받아 에디터의 각 버전에 맞게 덮어쓰기
2. 보안 적용 사이트 -> 패키지 설정 - Unity 암호화 옵션 활성화



3. 1 번 과정을 통해 소스 코드 삽입된 빌드를 보안 적용 사이트에 업로드하여 메타 데이터 암호화 적용된 앱 다운로드

### 주의

1. IL2CPP 로 설정된 Android 에서 적용 가능하며 iOS 에서는 il2cpp 관련 Precompiled 된 라이브러리를 사용하기 때문에 적용할 수 없음.
2. 복호화에 사용되는 meta\_key 는 유출 등의 이유로 요청 시 변경 가능함.

### OS 별 적용 파일 경로

#### Windows - Unity 2019

설치경로\Unity\Hub\Editor\버전\Editor\Data\il2cpp\libil2cpp\il2cpp-metadata.h

설치경로\Unity\Hub\Editor\버전\Editor\Data\il2cpp\libil2cpp\vm\MetadataLoader.cpp

#### Windows - Unity 2020 이상

설치경로\Unity\Hub\Editor\버전\Editor\Data\il2cpp\libil2cpp\vm\GlobalMetadataFileInternals.h

설치경로\Unity\Hub\Editor\버전\Editor\Data\il2cpp\libil2cpp\vm\MetadataLoader.cpp

#### MacOS - Unity 2019

/Applications/Unity/Hub/Editor/버전/Unity.app/Contents/il2cpp/libil2cpp/il2cpp-metadata.h

/Applications/Unity/Hub/Editor/버전/Unity.app/Contents/il2cpp/libil2cpp/vm/MetadataLoader.cpp

#### MacOS - Unity 2020 이상

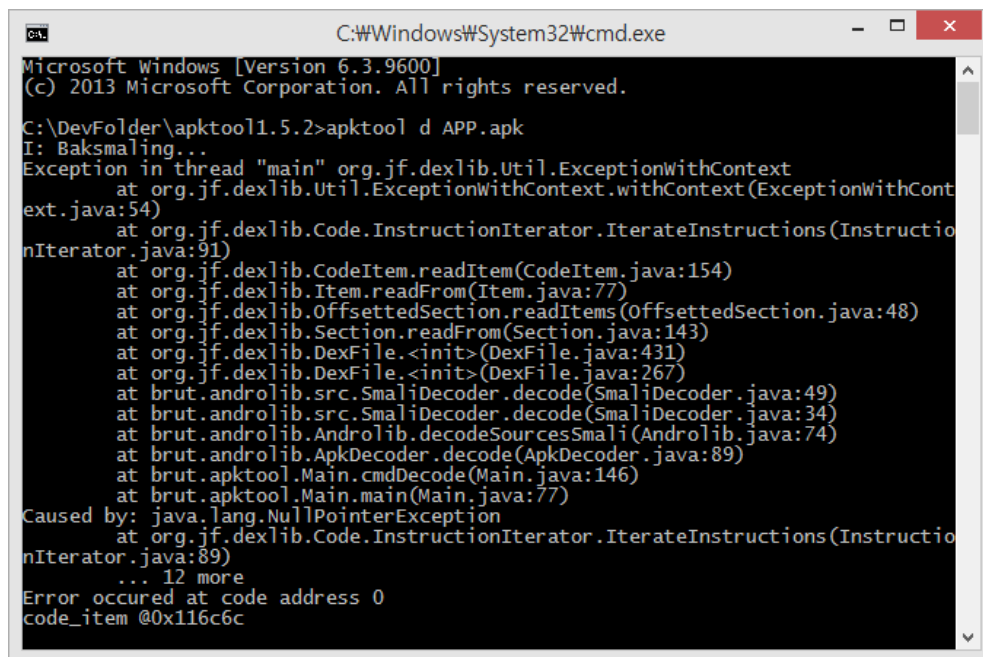
/Applications/Unity/Hub/Editor/버전/Unity.app/Contents/il2cpp/libil2cpp/vm/GlobalMetadataFileInternals.h

/Applications/Unity/Hub/Editor/버전/Unity.app/Contents/il2cpp/libil2cpp/vm/MetadataLoader.cpp

## G-PRESTO 테스트

### G-PRESTO 난독화 정상 적용 테스트

난독화 정상 적용 테스트를 위해 apktool 또는 dex2jar 등의 툴을 사용하여 정상적으로 디컴파일 되지 않는지에 대해 테스트를 진행할 수 있음.

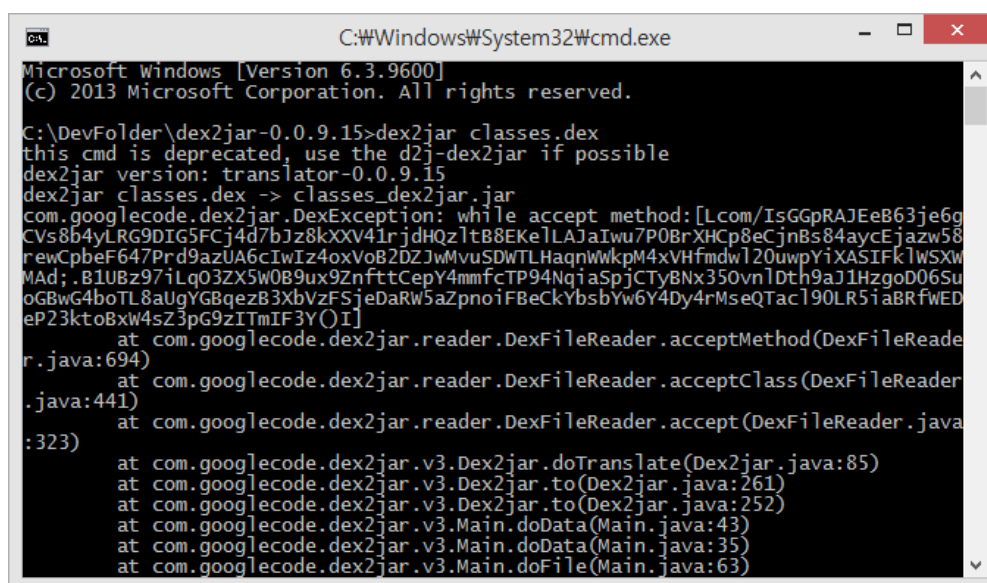


```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\DevFolder\apktool1.5.2>apktool d APP.apk
I: Baksmaling...
Exception in thread "main" org.jf.dexlib.Util.ExceptionWithContext
    at org.jf.dexlib.Util.ExceptionWithContext.withContext(ExceptionWithCont
ext.java:54)
    at org.jf.dexlib.Code.InstructionIterator.IterateInstructions(Instructio
nIterator.java:91)
    at org.jf.dexlib.CodeItem.readItem(CodeItem.java:154)
    at org.jf.dexlib.Item.readFrom(Item.java:77)
    at org.jf.dexlib.OffsettedSection.readItems(OffsettedSection.java:48)
    at org.jf.dexlib.Section.readFrom(Section.java:143)
    at org.jf.dexlib.DexFile.<init>(DexFile.java:431)
    at org.jf.dexlib.DexFile.<init>(DexFile.java:267)
    at brut.androlib.src.SmaliDecoder.decode(SmaliDecoder.java:49)
    at brut.androlib.src.SmaliDecoder.decode(SmaliDecoder.java:34)
    at brut.androlib.Androlib.decodeSourcesSmali(Androlib.java:74)
    at brut.androlib.ApkDecoder.decode(ApkDecoder.java:89)
    at brut.apktool.Main.cmdDecode(Main.java:146)
    at brut.apktool.Main.main(Main.java:77)
Caused by: java.lang.NullPointerException
    at org.jf.dexlib.Code.InstructionIterator.IterateInstructions(Instructio
nIterator.java:89)
    ... 12 more
Error occurred at code address 0
code_item @0x116c6c
  
```

<apktool 로 디컴파일 실패 화면>



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\DevFolder\dex2jar-0.0.9.15>dex2jar classes.dex
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar classes.dex -> classes_dex2jar.jar
com.googlecode.dex2jar.DexException: while accept method:[Lcom/IsGGpRAJEeB63je6g
CVs8b4yLRG9DIG5FCj4d7b2z8kXXV41rjdHQzltB8EKelLAJaIwu7P0BrXHCp8eCjnBs84aycEjazw58
rewCpbeF647Prd9azUA6cIwIz4oxVoB2DZJwMvuSDwTLHaqnWwkpM4xVHfmdw120uwpYiXASIFk1wSXW
MAd;.B1UBz97iLQ03ZX5W0B9ux9ZnfttCepY4mmfCTP94NqiaSpjCTyBNx350vn1Dth9aJ1Hgz0D06Su
oGBwG4boTL8AuGYGBqezB3xbVzFSjeDaRW5aZpnoiFBcKysbYw6Y4Dy4rMseQTac190LR5iaBRfWED
eP23ktoBxw4sZ3pG9zITmIF3Y()I]
    at com.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReade
r.java:694)
    at com.googlecode.dex2jar.reader.DexFileReader.acceptClass(DexFileReader
.java:441)
    at com.googlecode.dex2jar.reader.DexFileReader.accept(DexFileReader.java
:323)
    at com.googlecode.dex2jar.v3.Dex2jar.doTranslate(Dex2jar.java:85)
    at com.googlecode.dex2jar.v3.Dex2jar.to(Dex2jar.java:261)
    at com.googlecode.dex2jar.v3.Dex2jar.to(Dex2jar.java:252)
    at com.googlecode.dex2jar.v3.Main.doData(Main.java:43)
    at com.googlecode.dex2jar.v3.Main.doData(Main.java:35)
    at com.googlecode.dex2jar.v3.Main.doFile(Main.java:63)
  
```

<dex2jar 로 디컴파일 실패 화면>

## G-PRESTO 테스트

Apktool 관련 참고 사이트

<http://code.google.com/p/android-apktool/downloads/detail?name=apktool1.5.2.tar.bz2>

<http://choboitstory.tistory.com/151>

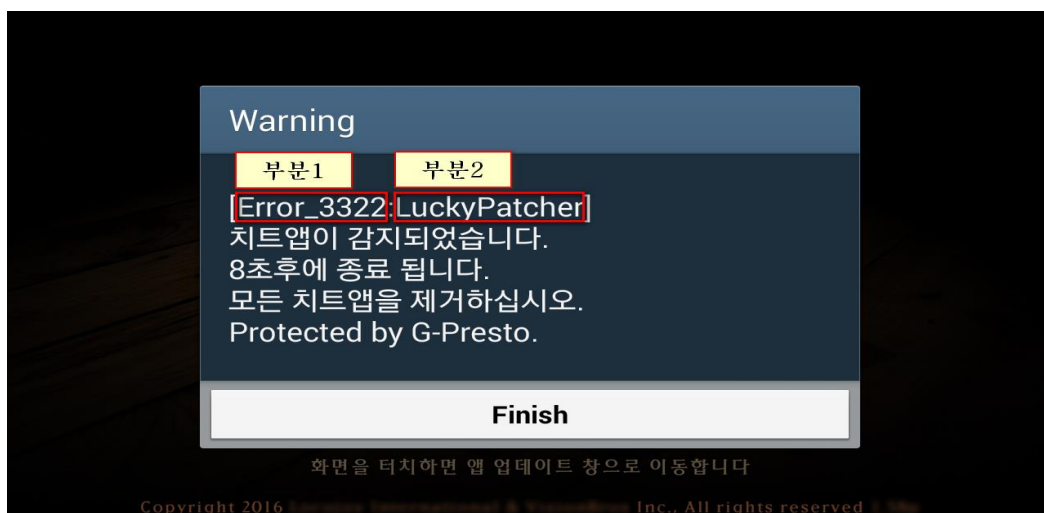
Dex2jar 관련 참고 사이트

<https://dex2jar.googlecode.com/files/dex2jar-0.0.9.15.zip>

<http://biig.tistory.com/11>

## G-PRESTO 치트앱 차단 테스트

메모리 해킹으로 많이 사용되는 GameGuardian, LuckyPatcher 등의 툴을 단말기에 설치한 상태에서 Presto 적용된 게임을 실행시 아래와 같이 차단 화면이 나오며 8 초후에 앱이 종료됨.

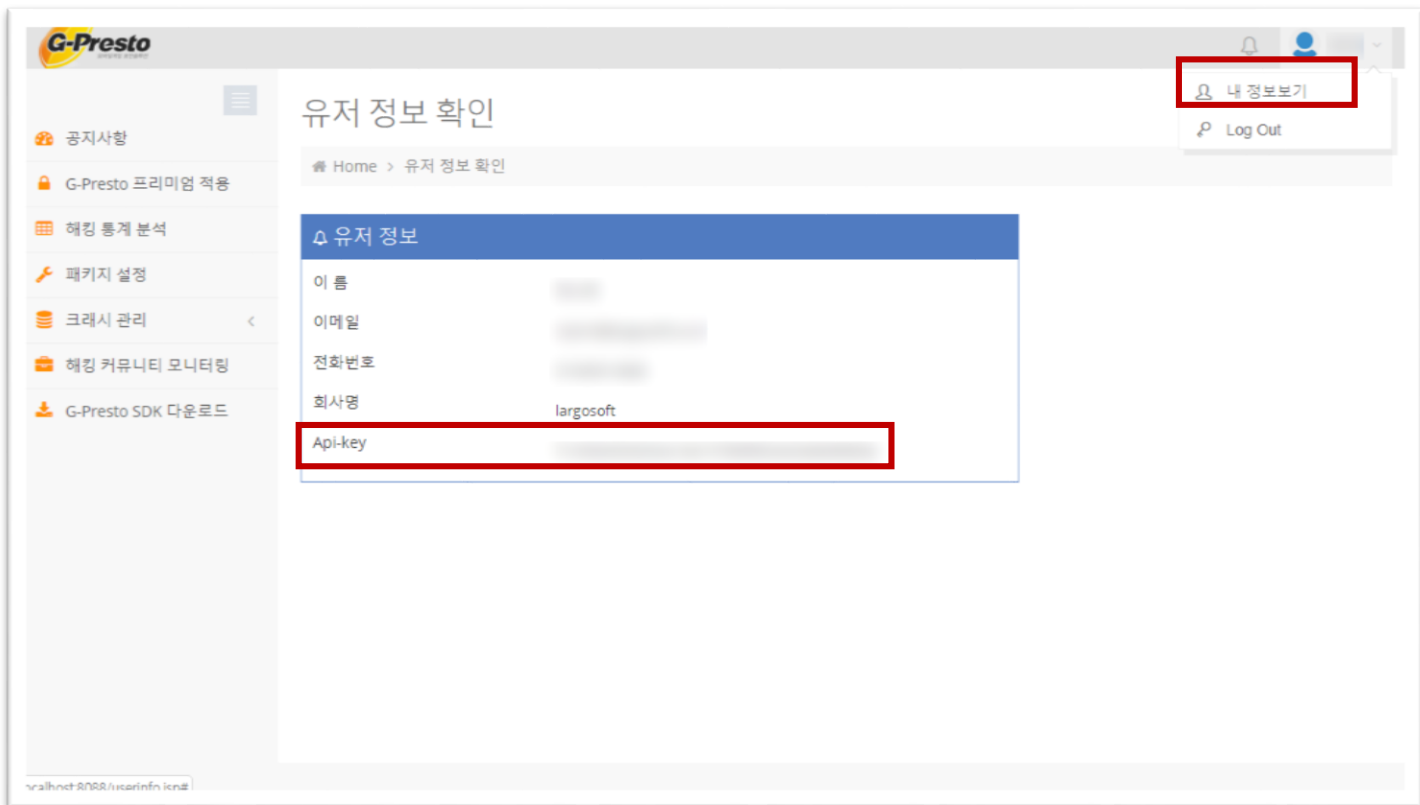


※ 앱이 종료되는 문제로 고객 문의 시 차단 화면에서 <부분 2>앱이름을 가진 앱 삭제를 안내하여 대응.



## G-PRESTO 사용자 정보 확인

### 사용자 정보 확인



<내 정보보기 클릭 화면>

G-PRESTO 사이트 우측상단의 "내 정보보기" 메뉴를 통해 사용자정보를 확인 가능

Api-key 의 경우 G-PRESTO Log Data Export 에 필요한 user\_authkey 가 됨

## G-Presto API

### G-PRESTO 가 제공하는 API (GETEMUL)

실행한 디바이스가 에뮬레이터인지 반환하는 API 입니다. 사이트에서 차단 및 로그를 확인 가능하지만 에뮬레이터 사용 여부 값을 별도로 처리하고자 할 때 호출하여 사용 할 수 있습니다.

G-Presto 에서 모니터링하고 있는 앱 에뮬레이터를 기준으로 판단됩니다.

보안 적용이 되지 않을 경우 동작하지 않습니다.

Unity 에서 호출 예)

```
AndroidJavaClass cls = new AndroidJavaClass ("com.bishopsoft.Presto.SDK.Presto");  
int isEmul = cls.CallStatic<int>("getEmul");  
//Debug.Log("isEmul = " + isEmul);
```

Cocos2d-x 호출 예)

```
#include "cocos2d.h"  
#include "platform/android/jni/JniHelper.h" //coco2d 에 기본 제공되는 JniHelper 를 이용  
  
JniMethodInfo jniMI;  
JniHelper::getStaticMethodInfo(jniMI, "com/bishopsoft/Presto/SDK/Presto" // G-Presto.getsEmul()위치  
                                , "getsEmul" //호출할 함수이름  
                                , "()Ljava/lang/Integer;"); // getsEngine 함수 타입  
jint isEmul = (jint) jniMI.env->CallStaticObjectMethod(jniMI.classID, jniMI.methodID); //getsEmul() 호출  
  
jniMI.env->DeleteLocalRef(jstr);  
//LOGD("%d",isEmul);
```

G-Presto 사이트에서 G-Presto 보안 적용-난독화 적용메뉴를 통해 보안을 적용하시면 자동으로 G-Presto 엔진 실행 코드가 앱에 삽입되어 실행되고 isEmul 를 가져 올 수 있습니다.

## G-PRESTO Log Data

### G-PRESTO LOG DATA 참고사항

G-PRESTO 는 접속로그, 블랙리스트 데이터를 JSON 형태로 제공

1. 사전에 상기 UUID 생성 방법을 통해 사용자에게 대한 UUID 를 서버에 저장하여야 UUID 매칭을 통해 사용자를 구분 가능
2. 로그 조회 범위 최대 5 일 ( from\_date 와 to\_date 차이가 5 일 이상이면 5 일치 데이터만 조회 )
3. 블랙리스트 조회 범위 최대 7 일
4. Data 조회에 사용되는 user\_authkey 는 사이트 우측 상단의 "내 정보보기"메뉴에서 확인 가능
5. 블랙리스트는 최근 7 일 동안 동일한 사용자의 시도 횟수 순으로 정보를 제공

### G-PRESTO Log Data JSON 응답 코드 형식

응답코드 형식 ( Json )

```
{
  "status": 1, //응답코드 - 0:error, 1:success, 2:checking
  "result": { //결과데이터 },
  "error": null, //에러코드
  "timestamp": 1392722292, //처리시작시간 - timestamp
  "duration": 0 //처리시간 - timestamp
}
```

Result 를 통해 결과를 확인

### G-PRESTO LOG DATA 요청

Log Data 날짜 지정 조회

[https://console.largosoft.co.kr/Presto\\_Data?type=logs&user\\_id=유저아이디&user\\_authkey=발급받은 user\\_authkey&from\\_date=20230501&to\\_date=20230503](https://console.largosoft.co.kr/Presto_Data?type=logs&user_id=유저아이디&user_authkey=발급받은 user_authkey&from_date=20230501&to_date=20230503) (조회할 범위 입력형식)

## G-PRESTO Log Data

### G-PRESTO BLACKLIST 요청

Blacklist 최근 7 일 조회

```
https://console.largosoft.co.kr/Presto_Data?type=blacklist&user_id=유저아이디&user_authkey=발급받은 user_authkey
```

Blacklist 특정 유저의 데이터 조회

```
https://console.largosoft.co.kr/Presto_Data?type=blacklist&user_id=유저아이디&user_authkey=발급받은 user_authkey&uuid=특정유저의 UUID
```

Blacklist 날짜 지정 조회

```
https://console.largosoft.co.kr/Presto_Data?type=blacklist&user_id=유저아이디&user_authkey=발급받은 user_authkey&from_date=20230501&to_date=20230503 (조회할 범위 입력형식)
```

Blacklist 해킹 시도 기준 설정 조회

```
https://console.largosoft.co.kr/Presto_Data?type=blacklist&user_id=유저아이디&user_authkey=발급받은 user_authkey&count=설정할 기준(1~100)
```

### G-PRESTO 블랙리스트 Data 요청 참고사항

블랙리스트 로그는 해킹 통계분석에서 제공하고 있으나,

G-Presto 블랙리스트 기준을 적용하여 매일 추출한 데이터를 해킹 통계분석에서 표시하기 때문에 기준치가 게임 사정책과는 다를 수 있습니다. Count 를 설정 하지 않을 경우 기준은 15 입니다.

날짜를 지정하여 해킹 시도 기준치에 따라 시도한 블랙리스트를 생성이 가능합니다.

G-presto 기준치 보다 낮게 설정하여 블랙리스트를 추적할 수 있습니다.

## G-PRESTO Log Data

### G-PRESTO 해킹커뮤니티 배포 앱 사용자 DATA 요청

해킹커뮤니티 배포 앱 사용자 로그 최근 15 일 조회

```
https://console.largosoft.co.kr/Presto_Data?type=cracklist&user_id=유저아이디&user_authkey=발급받은 user_authkey
```

해킹커뮤니티 배포 앱 사용자 로그 특정 데이터 조회

```
https://console.largosoft.co.kr/Presto_Data?type=cracklist&user_id=유저아이디&user_authkey=발급받은 user_authkey&uuid=특정유저 1 의 UUID1, 특정유저 2 의 UUID2
```

해킹커뮤니티 배포 앱 사용자 로그 패키지 및 날짜 지정 조회 (최대 15 일)

```
https://console.largosoft.co.kr/Presto_Data?type=cracklist&user_id=유저아이디&user_authkey=발급받은 user_authkey&from_date=20230501&to_date=20230503 (조회할 범위 입력형식)&pkg_name=패키지명 1,패키지명 2
```

### G-PRESTO 해킹커뮤니티 배포 앱 사용자 Data 요청 참고사항

해킹 커뮤니티 배포 앱 사용자 로그는 앱 실행 후 수분 뒤에 동작하기 때문에 앱 로그인 단계를 지난 다음 수집됩니다.

실시간으로 게임 서버 로그인 단계에서 차단을 위해 해킹 커뮤니티 배포앱 사용자 로그 API 를 호출하지 마시고,

해킹커뮤니티 배포 앱을 사용한 유저 리스트 생성을 위한 용도로 API 요청을 보내고,

G-PRESTO UUID 와 매칭되는 게임 계정 아이디로 유저 리스트를 구성하여 차단하시면 됩니다.

## G-PRESTO 기능 참고 사항

### G-Presto 적용 용량 변화

구글 플레이에서 프로젝트의 API level 13 이하는 50MB 용량 제한으로 적용에 따른 용량 증가를 미리 확인 필요

### 기본 엔진 크기

	G-Presto 적용 후 증가	비고
<b>엔진</b>	약 7.4MB	ABI 별 엔진 파일 전체 크기
<b>패턴 DB</b>	약 10kb	필수
<b>Java 난독화</b>	약 100kb-300kb	필수(Java 소스의 크기에 따라 변동)

### 추가 지원 여부에 따른 크기

	G-Presto 적용 후 증가	비고
<b>X86 지원</b>	약 660kb	지원 여부에 따라 반영
<b>Unity 암호화</b>	거의 변화 없음	옵션 적용 여부에 따라 반영

## G-PRESTO 기능 참고 사항

### 크로스 체크 기능 설명

사용자가 게임 접속 시 개발사 인증서버와 G-Presto 인증서버로 동시에 1 회성 키 값을 전송하여 해당 앱의 무결성을 확인하여 게임 서버와의 지속적인 통신을 진행 시키는 기능으로 추가적인 보안이 요구되는 경우 사용되며 게임 서버와 클라이언트에 본 개발사에서 제공되는 코드를 개발자가 추가해야 함

### 리소스 포인터 암호화(Res 암호화) 기능 설명

소스의 문자열 등 일부를 암호화하여 G-Presto 엔진을 통해 복호화 하게 되는데 G-Presto 엔진을 제거하거나 정상적으로 동작하지 않으면 정상적으로 복호화 되지 못하게 막기 때문에 G-Presto 엔진과의 결합도를 향상 시키는 기술로 보안이 향상되지만 보안 적용 시간이 길어지고 적용 이후 더 많은 게임 실행 테스트를 필요로 하기 때문에 초기 적용에서보다는 보안 이슈가 있으면 적용하는 것을 권장(오프라인 접속 가능한 경우가 아니라면 크로스체크 기능을 사용)

### 패키지 설정 옵션 변경

치트툴차단	앱위변조차단	가상머신차단	매크로차단	가상화앱차단	루팅가상머신차단	루팅가상화앱차단	Dex난독화	Res암호화	Unity암호화	루팅차단
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF

옵션 변경 실시간 반영

옵션 변경 후  
G-Presto 보안 재적용 시 반영

### <패키지 설정 옵션 변경>

#### 실시간 반영 옵션

- 차단 옵션을 변경하면 모든 유저가 앱을 다시 시작할 때 변경 사항이 반영
- 루팅 차단, 치트툴 차단, 앱위변조차단, 가상머신차단, 매크로차단, 가상화앱차단, 루팅 가상화머신차단, 루팅가상화앱 차단

#### 보안 재적용시 반영 옵션

- 옵션 변경 후 G-Presto 보안 적용 메뉴에서 앱을 업로드하면 옵션이 반영
- dex 난독화, Res 암호화, unity 암호화

## G-PRESTO 기능 참고 사항

### 빌드툴 통합

Jenkins 같은 빌드툴에 통합하기 위해 curl 를 통해 apk 파일을 업로드하고 G-Presto 적용된 apk 를 처리 가능.

#### 파일 업로드 예제

linux

```
curl -F -file=@파일명.apk 'https://console.largosoft.co.kr/Upload?user_id=유저아이디&mode=a'
```

Windows

```
curl -F -file=@파일명.apk "https://console.largosoft.co.kr/Upload?user_id=유저아이디&mode=a"
```

#### 응답값 예제

```
[{"link":"W/WORKW/20180920141445W/파일명.apk",  
"fullpath":"https://W/W/console.largosoft.co.krW/WORKW/20180920141445W/파일명.apk","success":true}]
```

success : 성공 여부

link : 서버 주소를 포함하지 않은 서버 내에서 파일의 위치

fullpath : 서버 주소를 포함한 파일 다운 로드 경로



## G-PRESTO 기능 참고 사항

Jenkins 에서 Pipeline 과 Pipeline Utility Steps 플러그인을 이용하여 파일 업로드와 다운로드를 처리하는 과정

Jenkins 연동은 업체별로 적용 방식이 상이 합니다. Jenkins 스크립트로 가능한 예제이며 외부 다운로드 프로그램을 만들어서 사용하는 경우도 있습니다. 참고 바랍니다.

간단한 pipeline 코드 예제

```
pipeline {
  agent any
  stages {
    stage('Hello') {
      steps {
        script{
          def response = bat(script:'curl -F -file=@파일명.apk
            "https://console.largosoft.co.kr/Upload?user_id=유저아이디&mode=a"', returnStdout: true).trim()
          //bat 실행 명령 문 제거
          def responseParsed = response.readLines().drop(1).join(",")
          //json 전환
          def props = readJSON text: responseParsed
          echo props[0]['fullpath']
          //추출한 링크로 다시 요청
          response =
            bat(script:"curl " + props[0]['fullpath'] + " --output result.zip", returnStdout: true).trim()
        }
      }
    }
  }
}
```

Pipeline Utility Steps : <https://www.jenkins.io/doc/pipeline/steps/pipeline-utility-steps/#readjson-read-json-from-files-in-the-workspace>

# 안드로이드 스토어 App Signing 관련 변경 사항 안내

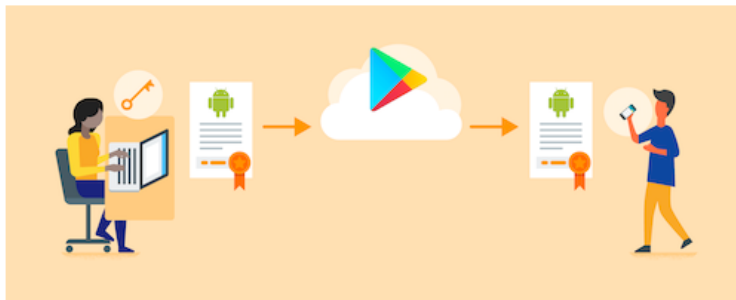
## ANDROID STORE APP SIGNING 관련 변경 사항 안내

### 앱 서명 키 관리

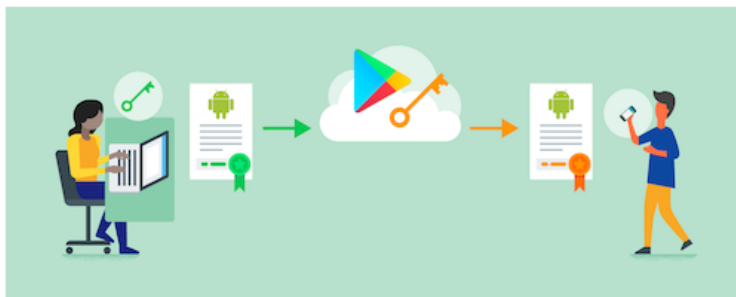
Google Play App Signing을 사용하면 새로운 앱이나 기존 앱의 앱 서명 키를 안전하게 관리할 수 있습니다. 서명 키는 Google에서 자체 키를 저장하는 데 사용하는 것과 같은 보안 인프라에 저장됩니다.

### Google Play App Signing 정보

*Google Play App Signing 사용하지 않음:* 앱 서명 키를 사용하여 앱에 서명한 다음 앱을 Google Play에 업로드하면 앱이 사용자에게 제공됩니다.



*Google Play App Signing 사용:* 업로드 키를 사용하여 앱에 서명합니다. 그런 다음 Google에서 업로드 키 서명을 인증한 후 삭제합니다. 마지막으로 Google에서 내가 제공한 원래 앱 서명 키를 사용하여 앱에 다시 서명하고 사용자에게 앱을 제공합니다.



구글 플레이 콘솔에서 앱을 등록하게 되면 Google Play App Signing 기능을 사용하도록 유도함.

ONE store 에서도 ONE store App 서명 사용을 사용하도록 유도

Google Play 관련 문서

<https://support.google.com/googleplay/android-developer/answer/9842756?hl=ko#>

ONE store 관련 문서

<https://dev.onestore.co.kr/wiki/ko/doc/one-store-8292368.html>

## 안드로이드 스토어 App Signing 관련 변경 사항 안내

Google Play App Signing, 원 스토어 앱 서명(ONE store 가 앱 서명키를 관리 보호) 적용 시 스토어 업로드 이후에 앱 무결성 정보가 변경됩니다.

‘패키지 설정’ 메뉴에서 안드로이드 마켓을 대상으로 하는 앱은 마켓 무결성 정보를 확인 하기 전까지 앱 위변조 차단 옵션이 일시적으로 비 활성화됩니다.

무결성 정보를 등록한 이 후 앱 위변조 차단 옵션이 활성화 가능합니다.

마켓 무결성 정보를 확인하기 위해서는 G-Presto 가 적용된 앱을 마켓 채널에 업로드 후 실행해야 합니다. 이하 방법을 참고하시어 편한 방법으로 관리자에게 전달해주시기 바랍니다.

무결성 정보 등록하기 위한 방법 안내

- 플레이스토어 테스트 채널에 업로드를 거친 앱을 다운받아 전달
- 원 스토어 바이너리 파일을 다운받아 전달
- 플레이스토어 테스트 채널에 G-Presto 관리자 계정을 테스트 계정으로 등록
- 스토어 테스트 채널에 업로드를 거친 앱을 실행한 기기 정보 및 시간 전달
- 원 스토어 바이너리에서 다운 받은 앱을 실행한 기기 정보 및 시간 전달

Google Play Console 에서 Google Play App Signing 적용 여부 확인 방법

출시관리 - 앱서명

 이 앱에 Google Play App Signing이 사용 설정되었습니다. [자세히 알아보기](#)

ONE store 에서 앱 서명 적용 여부 확인방법

신규 바이러니 (상용 바이러니) - 바이너리 - 서명키

서명키

[ONE store가 앱 서명키를 관리 보호](#)

## 안드로이드 앱 번들

### ANDROID APP BUNDLE

#### App Bundle 적용 고려 이유

2019 년 8 월 1 일부터 Google Play 에 게시되는 앱에서는 64 비트 아키텍처를 지원해야 합니다. 구글 플레이에 등록 가능한 APK 용량은 100MB 로 제한되어 있기 때문에 많은 앱들이 64 비트 아키텍처를 지원할 경우 제한 용량을 초과하는 문제가 발생 됩니다. 용량 제한 문제를 해결하고 추후 구글 정책이 앱 번들 사용 강제 할 경우를 대비해서 미리 적용 테스트가 필요 합니다.

#### App Bundle 이란?

App Bundle 이란 APK 와 비슷하지만 모든 코드, 리소스, CPU 아키텍처와 메타데이터를 압축한 zip 파일입니다.

그래서 Google Play 는 App Bundle 에서 사용자 기기에 필요한 코드와 리소스만을 선택해 빌드 될 수 있는 것입니다.

App Bundle 을 빌드하면 .aab 파일이 생성됩니다.

빌드된 aab 파일을 Play Store 에 업로드하면 Play Store 가 각각 기기에 최적화된 APK 를 빌드합니다.

Dynamic Delivery 란 사용자 기기에 필요한 리소스만을 다운로드를 가능하게 해주는 구글의 워크플로우 입니다.

Dynamic Delivery 는 Android 5.0(SDK 21) 이상부터 사용할 수 있으며, Split APK 메커니즘을 이용한다고 합니다.

즉, 분리된 APK 를 하나의 앱으로 만들어주는 것으로 여러 가지 기능을 분리하고 나중에 기능이 필요로 할 때 다운로드 받아 설치됩니다.

따라서 Dynamic Delivery 로 Google Play 는 각 기기의 맞는 리소스만 빌드를 하고 기기에 설치되도록 한 다음에 사용자가 특정 기능을 필요로 할 때 부분적으로 다운받아서 사용할 수 있게 해주기 때문에 앱 용량을 줄일 수 있습니다.

#### App Bundle 업로드의 이점

App Bundle 을 사용하면 광범위한 기기 설정에 최적화된 APK 를 지원하기 위해 하나의 아티팩트만 빌드, 서명, 업로드하면 됩니다. 그러면 Google Play 에서 앱의 APK 를 대신 관리하고 게시합니다. 따라서 지원하려는 ABI, 화면 밀도 및 언어 조합별로 버전 코드를 관리할 필요가 없습니다. 또한 App Bundle 을 사용하면 게시 프로세스에 지속적으로 추가되는 개선사항의 혜택을 받을 수 있습니다.

다운로드 크기와 디스크 할당 크기가 작아집니다.

사용자의 기기 대신 APK 에 저장되는 압축되지 않은 기본 라이브러리를 사용하여(Android 6.0 이상) 다운로드 크기, 디스크 할당 크기, 설치 시간을 줄일 수 있습니다.

사용자에게 필요한 기능 및 설정을 설치 중이 아닌 사용자가 요청할 때 제공할 수 있습니다.

여러 개의 APK 를 빌드하고 게시할 필요가 없어 빌드 및 출시 관리가 간단해집니다.

Play Console 에 App Bundle 을 업로드하면 Google Play 에서 기기에 최적화된 바이너리를 전송합니다.

Android 5.0 이상: Play 가 기본 APK, 설정 APK 및 동적 기능 APK(해당하는 경우)를 생성합니다.

Android 5.0 미만: Play 가 서버 측에서 멀티 APK 를 생성합니다.

## 안드로이드 앱 번들

### App Bundle 적용 테스트 확인 사항

G-Presto 사이트에서 보안 적용 이후 구글 플레이 콘솔에서 내부, 비공개(알파), 공개(베타) 테스트 중 가능한 테스트 방식을 적용하여 사전 테스트 후 정식 출시 하시기 바랍니다.

디바이스 환경에 따라 armeabi-v7a, arm64-v8a, x86, x86\_64 네이티브 라이브러리가 포함되는 경우가 다르기 때문에 각각의 아키텍처를 지원하는 디바이스에서 테스트가 이루어져야 합니다.

### 관련 참고 사이트

<https://developer.android.com/guide/app-bundle>

<https://zerogdev.blogspot.com/2018/07/android-app-bundle.html>

<https://developer.android.com/distribute/best-practices/develop/64-bit>

## 구글 정책 변경 안내

### ANDROID 11 이상의 QUERY\_ALL\_PACKAGES 권한 요구

구글 플레이 정책 변경으로 인하여 Android 11(API 30) 이상을 타겟으로 하는 앱의 경우 보안 엔진이 Android 11 환경에서 설치된 패키지 목록을 확인하는 과정에 설치된 앱 일부 목록만을 가져오게 됩니다. 따라서 모든 설치 앱 목록을 가져오기 위해서는 아래 사항과 링크를 참조하여 적용이 필수적으로 필요 합니다.

패키지 목록을 가져오기 위해서 AndroidManifest.xml 내부에 QUERY\_ALL\_PACKAGES 권한을 포함해야 합니다.

```
<uses-permission android:name="android.permission.QUERY_ALL_PACKAGES" />
```

**해당 권한이 적용된 앱은 Play Console 에서 권한 사용을 위한 양식을 작성하지 않을 경우**

**22 년 7 월 12 일 이후부터 앱 업데이트를 제출 할 수 없게 됩니다.**

**작성 양식 가이드에 따라 작성하시기 바랍니다.**

### Play Console 에서 권한 사용을 위한 양식 작성 방안 가이드

[https://console.largosoft.co.kr/guide\\_playconsole.jsp](https://console.largosoft.co.kr/guide_playconsole.jsp)

#### 미리보기: 폭넓은 패키지(앱) 가시성 (QUERY\_ALL\_PACKAGES) 권한 사용



이제 선언을 제출하여 QUERY\_ALL\_PACKAGES 권한을 사용할 수 있습니다.

선언하지 않으면 7월 12일부터는 앱 업데이트를 제출할 수 없게 됩니다. 7월 12일까지 앱을 업데이트해야 합니다.

Google Play에서는 기기에 설치된 앱 인벤토리를 확인할 수 있게 해 주는 QUERY\_ALL\_PACKAGES 권한을 포함해 위험성이 높거나 민감한 권한 [이](#)의 사용을 제한합니다. Play에서는 사용자 기기에서 처리되는 설치된 앱 인벤토리를 개인 정보 및 민감한 정보로 간주하며, 권한 사용은 앱의 사용자 대상 기능 또는 목적을 위해 사용자 기기에 설치된 앱에 관한 폭넓은 가시성이 필요한 경우에만 허용됩니다.

앱이 아래의 허용되는 용도 요구사항을 충족하지 않는 경우 Play 정책을 준수하기 위해 앱의 매니페스트에서 권한을 삭제해야 합니다. 정책을 준수하는 대체 구현 방법에 관한 제안사항도 아래에 자세히 설명되어 있습니다.

앱이 QUERY\_ALL\_PACKAGES 권한의 허용되는 용도에 관한 요구사항을 충족할 경우, Play Console의 [권한 선언 양식](#) [이](#)를 사용해 이 권한 및 위험성이 높은 기타 모든 권한을 선언해야 합니다.

정책 요구사항을 충족하지 못하거나 권한 선언 양식을 제출하지 않으면 앱이 Google Play에서 삭제될 수 있습니다.

**중요:** 앱에서 제한된 권한을 사용하는 방식을 변경하려면 정확하게 업데이트된 정보로 요청을 수정해야 합니다. 이러한 권한을 사기성 및 선언되지 않은 용도로 사용하면 앱이 금지되거나 개발자 계정이 해지될 수 있습니다.

#### QUERY\_ALL\_PACKAGES 권한은 언제 요청해야 하나요?

QUERY\_ALL\_PACKAGES 권한은 앱이 Android 11 이상을 실행하는 기기에서 Android API 수준 30 이상을 타겟팅하는 경우에만 적용됩니다.

이 권한을 사용하려면 앱이 아래의 허용되는 사용 범위에 속해야 하며 기기의 모든 앱을 검색하기 위한 핵심 목적이 있어야 합니다. 범위가 좁은 앱 가시성 확보 방식이 앱의 정책을 준수하는 사용자 대상 핵심 기능을 충분히 지원하지 못하는 이유를 설명할 수 있어야 합니다.

핵심 기능은 앱의 주 목적으로 정의됩니다. 기기의 모든 앱을 검색하는 이 핵심 기능이 없으면 '기능이 부족한' 앱이거나 사용할 수 없게 됩니다. 핵심 기능 및 이러한 핵심 기능을 구성하는 모든 핵심 특징은 모두 앱 설명에서 두드러지게 소개 및 홍보되어야 합니다.

#### 패키지(앱) 가시성 권한

기기에서 처리되는 설치된 앱의 인벤토리는 [개인 정보 및 민감한 정보](#) [이](#) 정책이 적용되는 개인 데이터 및 민감한 사용자 데이터로 간주되며, 다음 요구사항이 적용됩니다.

기기 내 다른 앱의 실행, 검색 또는 다른 앱과의 상호 운용이 핵심 목적인 앱은 아래에 간략히 설명된 것처럼 적합한 범위 내에서 기기에 설치된 다른 앱을 확인할 수 있습니다.

- **광범위한 가시성:** 광범위한 가시성은 앱이 기기에 설치된 앱(패키지)을 폭넓게(또는 '광범위하게') 확인할 수 있는 기능입니다.
- **API 수준 30 이상** [이](#)를 타겟팅하는 앱의 경우, QUERY\_ALL\_PACKAGES [이](#) 권한을 통해 부여된 광범위한 가시성은 앱이 작동하기 위해 기기 내 모든 앱을 인지하거나 이와 상호작용해야 하는 특정 사용 사례로 제한됩니다.
- 앱이 [보위카 더 정확히 타겟팅된 패키지 가시성 선언](#) [이](#) 만으로도 작동 가능한 경우 QUERY\_ALL\_PACKAGES를 사용할 수 없습니다. 예를 들어, 광범위한 가시성을 요청하는 대신 특정 패키지를 처리하고 이와 상호작용하는 경우가 여기에 해당합니다.
- 대체 메서드를 사용하여 QUERY\_ALL\_PACKAGES 권한과 관련된 광범위한 가시성을 대략적으로 얻을 수 있지만 이 또한 사용자를 대상으로 한 핵심 앱 기능과 이 메서드를 통해 발견된 모든 앱과의 상호 운용으로 제한됩니다.
- QUERY\_ALL\_PACKAGES 권한이 허용되는 사용 사례를 알아보려면 [이 고객센터 도움말](#) [이](#)를 참고하세요.
- **제한된 앱 가시성:** 제한된 가시성은 앱이 더 정확하게 타겟팅된(광범위하지 않음) 메서드를 사용하여 특정 앱을 처리함으로써 데이터 액세스를 최소화합니다. 예를 들어, 앱의 매니페스트 선언에 맞는 특정 앱을 처리하는 경우가 여기에 해당합니다. 앱이 정책을 준수하는 방식으로 상호 운용되거나 이러한 앱을 관리하는 경우 이 메서드를 사용하여 앱을 처리할 수 있습니다.
- 기기에 설치된 앱의 인벤토리에 대한 가시성은 사용자 앱 내에서 액세스하는 핵심 기능이나 앱의 핵심 목적과 직접적인 관련이 있어야 합니다.

Play를 통해 배포된 앱에서 처리되는 앱 인벤토리 데이터는 분석 또는 광고 수익 창출 목적으로 판매되거나 공유되어서는 안 됩니다.

#### 관련 링크 :

구글 플레이 콘솔 고객센터 도움말 (미리보기: 폭넓은 패키지(앱) 가시성 (QUERY\_ALL\_PACKAGES) 권한 사용)

[https://support.google.com/googleplay/android-developer/answer/10158779?hl=ko&ref\\_topic=2364761](https://support.google.com/googleplay/android-developer/answer/10158779?hl=ko&ref_topic=2364761)

민감한 정보에 액세스하는 권한 및 API

[https://support.google.com/googleplay/android-developer/answer/9888170?hl=ko&ref\\_topic=9877467#pkg-app-visibility](https://support.google.com/googleplay/android-developer/answer/9888170?hl=ko&ref_topic=9877467#pkg-app-visibility)

## 자주 묻는 질문 (FAQ)

### 자주 묻는 질문 FAQ

#### 1. G-Presto의 치트 탐지 방식은 어떤 방식인가요?

- 기본적으로 설치된 앱 기반 검사와 프로세스 검사 방식으로 치트 툴을 탐지합니다.
- 설치만 되어 있어도 차단이 진행되며, 앱 실행 중에 설치되는 경우에도 차단합니다.
- Android 6 버전 이후 프로세스 접근 권한이 제한되어, Android 6 이상 버전에서는 프로세스 검사가 이루어지지 않습니다.
- 에뮬레이터 탐지 방식의 경우 커널 명과 같은 OS 정보와 시스템 라이브러리, 설치된 앱 등 수집 가능한 다양한 정보를 통해 복합적인 방식으로 탐지하고 있습니다.

#### 2. '모비즌'과 같은 미러링 앱은 차단이 되지 않나요?

- 미러링 앱은 악의적인 목적으로 사용되는 경우가 적고, 설치된 것으로만 차단하게 될 경우 유저들의 진입장벽이 될 수 있어서 현재 차단을 하고 있지 않습니다.

#### 3. 해킹통계분석에서 해킹 커뮤니티 배포 앱 사용자 로그에는 어떤 로그가 제공되나요?

- 앱 무결성에 관련된 정보와 추가된 라이브러리 등의 정보를 별도로 수집하여, 수집된 정보를 기반으로 게임사의 UUID와 매칭하여 유저를 특정하여 차단하고 있습니다.

#### 4. Android에서 실시간 차단 옵션으로 루팅 옵션이 제공되지 않나요?

- 루팅된 단말기를 사용하는 정상적인 유저가 차단되는 경우가 많아 사용자의 실수로 루팅을 차단하여 많은 사용자가 차단되는 문제의 발생을 막기 위해 웹에서 변경되지 않도록 되어 있습니다.
- 요청하시는 경우 차단 옵션을 적용 할 수 있습니다.
- 로그를 통해 루팅 수치와 해킹 시도율을 확인 후 적용하시도록 가이드 드리고 있습니다.

## 자주 묻는 질문 (FAQ)

5. G-Presto Unity plug-in 을 적용 후 빌드 용량이 늘어나게 되는데, 웹 업로드 및 난독화 적용 이후 빌드 용량이 줄어드는데 정상인가요?

- 정상입니다.
- apktool 을 통해 apk 가 재생성 되는 과정에서 압축률이 다르게 적용되는 것으로 확인 됩니다.

6. 에뮬레이터를 이용한 PC 매크로(녹스 마우스 녹화기능, 게임전용 매크로 등) 사용 시 탐지가 가능한가요?

- 프로세스나 설치된 앱의 패턴으로 검사하는 방식으로는 PC 매크로를 감지할 수 없습니다.
- 개발사에서 게임 중간에 퍼즐이나 'captcha' 를 사용하는 것을 권장 드리고 있습니다.

7. 디버깅 방지 기능은 ON/OFF 가 불가능 한가요? 또한 디버깅 시도 탐지에도 로그를 전송하나요?

- 디버깅 방지 기능 구조상 ON/OFF 기능은 제공하지 않고 있습니다.
- 접근한 프로세스명을 서버로 전송하며 메모리해킹툴 사용(MEMHACK)으로 서버에 저장 됩니다