

# G-PRESTO 사용 가이드

iOS 연동 및 테스트 메뉴얼

## 목차

기능 리스트 .....	3
UNITY 플러그인 .....	5
UNITY 플러그인 프로젝트 설정 .....	5
UNITY 플러그인 운영체제별 적용 .....	6
G-Presto iOS UNITY실행 코드 적용 .....	7
UNITY 플러그인 변수 암호화 사용 예제 .....	7
UNITY 플러그인 스피드핵 감지 사용 예제 .....	9
Unreal Engine 프로젝트 적용 .....	10
G-Presto Xcode에서 적용 .....	12
G-Presto 라이브러리 추가 .....	12
G-Presto iOS 실행 코드 적용 [objective-c] .....	13
G-Presto iOS 실행 코드 적용 [swift] .....	14
엔진 코드 bitcode 설정 .....	15
uuid를 통한 치트 유저 확인 .....	16
실시간 검사 옵션 적용 .....	17
탈옥 단말기 테스트 .....	18
인앱 결제 영수증 검증 .....	19
인앱 결제 확인 사항 .....	19
인앱 결제 영수증 검증 구현 참고 사이트 .....	19
크로스체크 Unity에서 함수 호출 .....	20
크로스체크 Unity이외 함수 호출 .....	20
주의 사항 .....	20
자주 묻는 질문 FAQ .....	21
문제 해결 .....	22

## G-PRESTO iOS 기능 소개

### 기능 리스트

#### 탈옥 체크

메모리 치트 툴이 설치 및 실행되기 위해서는 필수적으로 단말기를 탈옥하게 됩니다. 단말기의 탈옥 여부를 다양한 방법으로 확인하는 기능을 갖추고 있으며 지속적으로 추가하고 있습니다.

#### 디버깅 방지

단말기에 디버깅 기기 혹은 GDB 같은 디버거(debugger)를 이용해서 프로세스에 접근하여 코드 흐름을 조작하거나 메모리 값을 변조하는 것을 차단하는 기능을 갖추고 있습니다.

※참고 : xcode에서 실행시 디버깅 방지 코드로 인해서 앱이 실행 후 바로 종료될 수 있습니다. 필요시 디버깅 방지 기능이 제외된 엔진을 요청 주시기 바랍니다.

#### 프로세스 탐지

GameGuardian 등의 iOS 환경에서 동작하는 메모리 치트 툴의 프로세스의 실행 여부를 확인하는 기능을 갖추고 있습니다.

#### 앱위변조 탐지

앱의 다양한 정보를 확인하여 위변조 여부를 확인하는 기능을 갖추고 있습니다. G-Presto 엔진 자체를 실행되지 않도록 우회되지 않도록 크로스체크 기능 사용을 가이드 하고 있습니다.

#### 메모리 변조 체크

유니티 프로젝트의 경우 메모리 치트툴에 의해 메모리에 저장된 특정 값을 검색되지 않도록 방지하며 변조될 경우 앱을 종료하는 기능을 갖추고 있습니다.

## G-PRESTO iOS 기능 소개

### 엔진 코드 난독화

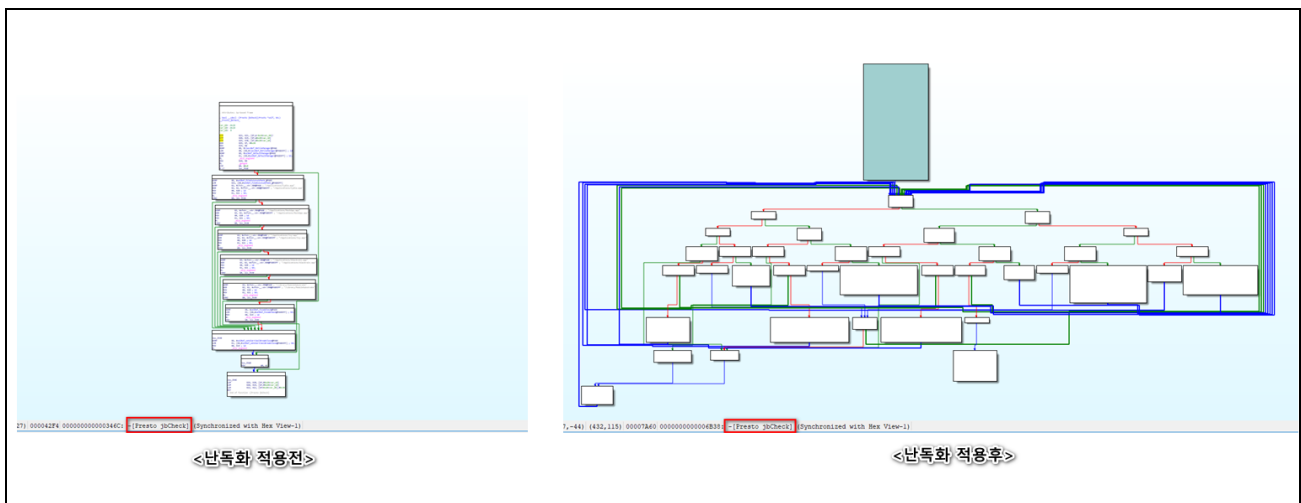
컴파일러 레벨에서 코드 난독화 기능을 추가하여 G-Presto 엔진 정적 분석을 어렵게 하는 난독화 기능을 갖추고 있습니다.

Bogus Control Flow – 가짜 코드를 삽입하여 흐름을 난독화 함.

Control Flow – 코드 중간에 새로운 코드 흐름을 삽입하여 복잡도를 증가시켜 난독화 함.

Flattening Instruction Substitution – 계산식을 변형하여 난독화 함.

String Obfuscation – 문자열을 암호화함.



<난독화 적용 전후 엔진 비교>

### 로그

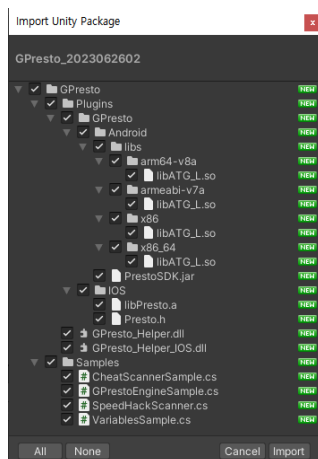
게임 내의 탈옥 및 치트 툴을 사용하는 유저의 정보를 확인할 수 있으며 통계를 제공합니다.

## G-PRESTO Unity 플러그인 적용

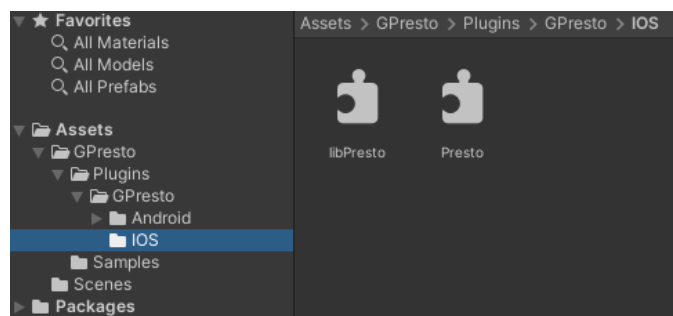
### UNITY 플러그인

G-Presto는 메모리 치트툴을 통한 변수의 메모리 주소를 찾는 행위를 방지하기 위해 Unity용 변수 암호화 플러그인을 제공합니다.

### UNITY 플러그인 프로젝트 설정



<Unity 플러그인 import 전>

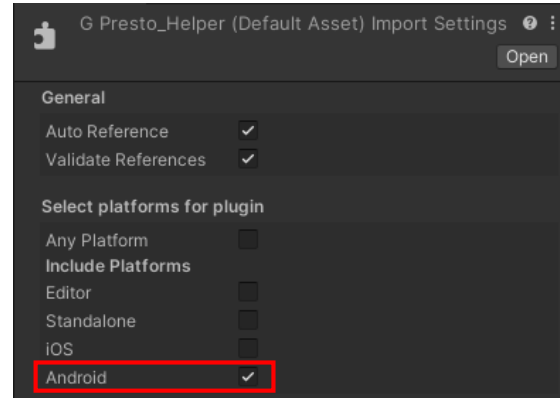
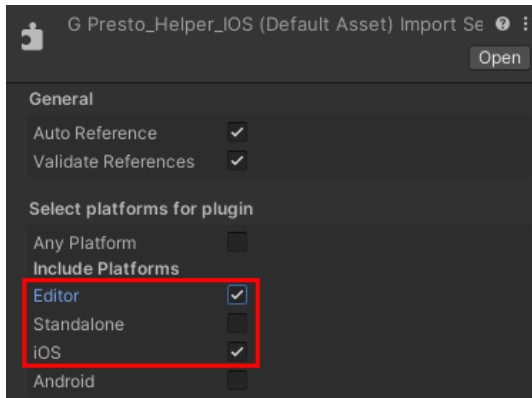


<Unity 플러그인 import 후>

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

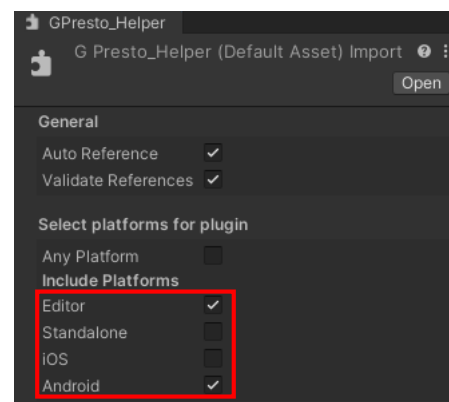
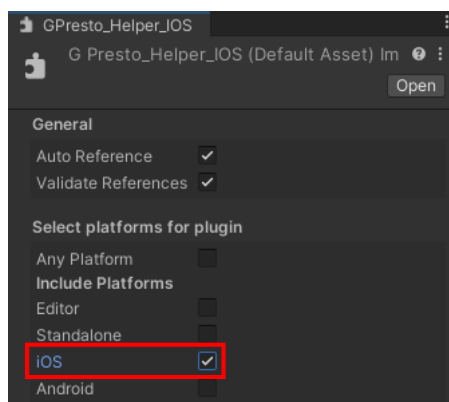
### UNITY 플러그인 운영체제별 적용



<iOS빌드의 경우>

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

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



<Android 빌드의 경우>

GPresto\_Helper.dll – Android, Editor 선택.

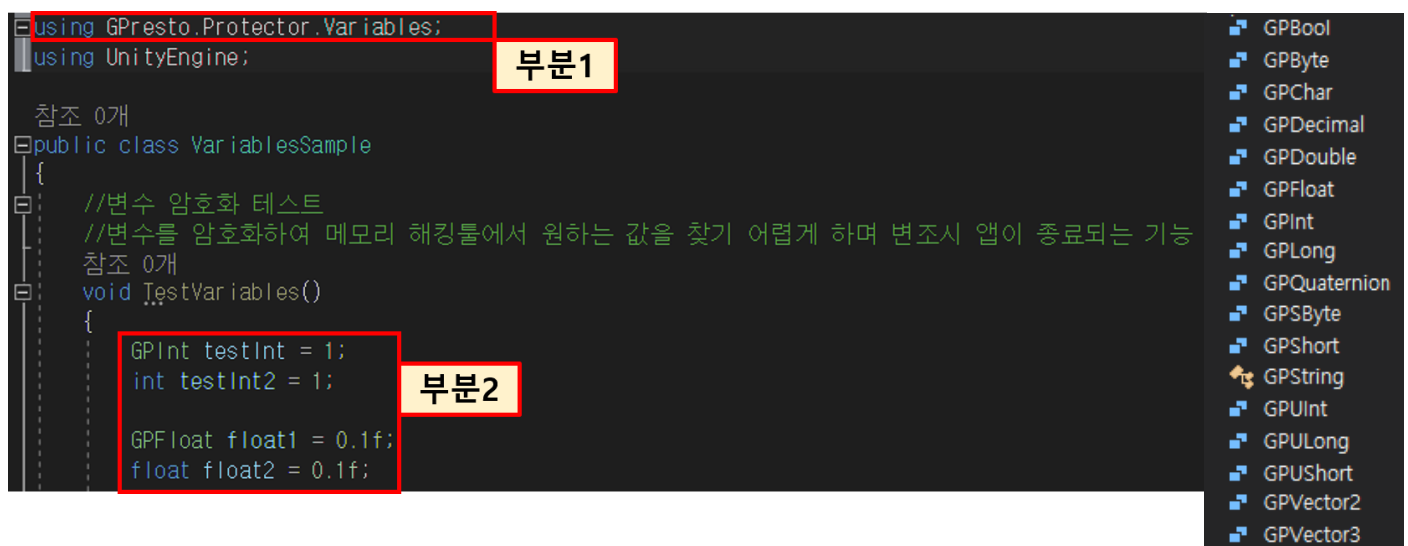
GPresto\_Helper\_IOS.dll – iOS 만 선택, editor는 선택하지 않음.

## G-PRESTO Unity 프로젝트 적용

### G-PRESTO IOS UNITY 실행 코드 적용

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

### UNITY 플러그인 변수 암호화 사용 예제



<Unity C#Script에서 GP자료형 선언과 사용가능한 GP자료형 종류>

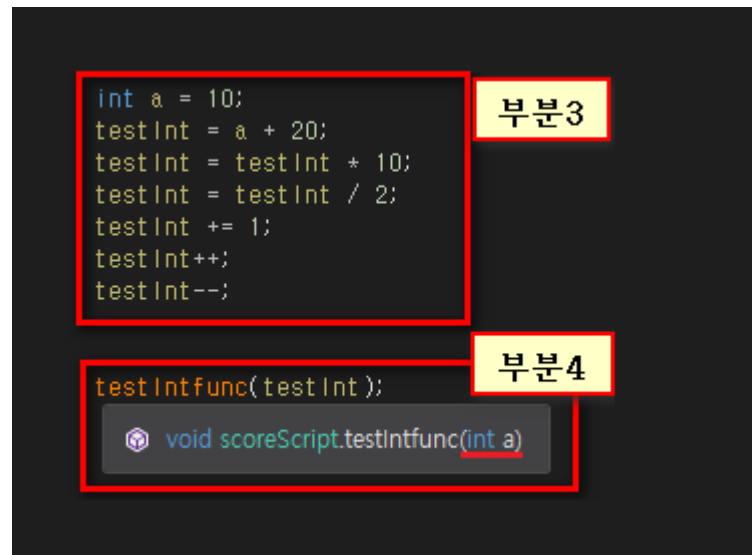
1. <부분1>

G-Presto 자료형을 사용하기 위하여 네임스페이스 추가

2. <부분2>

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

## G-PRESTO Unity 프로젝트 적용



<Unity C#Script GPlnt 사용 예>

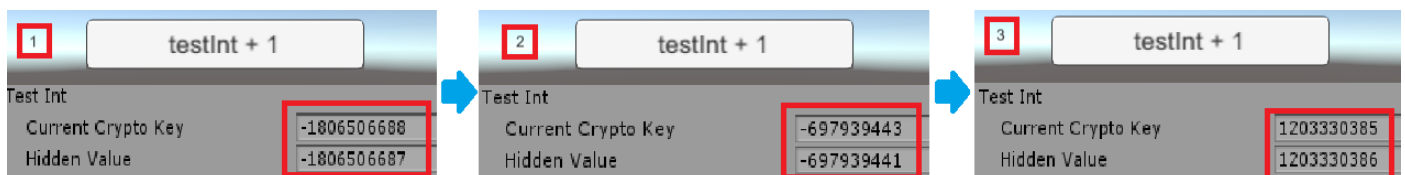
### 3. <부분3>

<부분2>에 선언된 GPlnt 자료형 사용 예시(GPlnt는 int와 같이 연산이 가능함.)

### 4. <부분4>

<부분2>에 선언된 GPlnt 자료형 사용 예시(int형을 매개변수로 하는 testIntfunc함수를 GPlnt형인 testInt 변수를 전달하여 호출가능 함.)

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



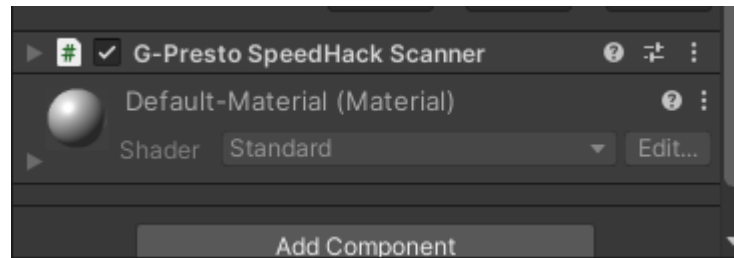
<GPlnt testInt의 암호화된 값>



## 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.cs를 Add Component – Script>

스피드해킹 차단 기능을 시작하고자 하는 게임 오브젝트에 SpeedHackScanner.cs를 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 SDK에서 제공하는 차단 화면을 출력하고 앱 종료됩니다.

**Unity 플러그인 기능의 차단 기능은 G-Presto 보안 적용이 요구됩니다. (libPresto.a 파일 빌드 포함 요구)**

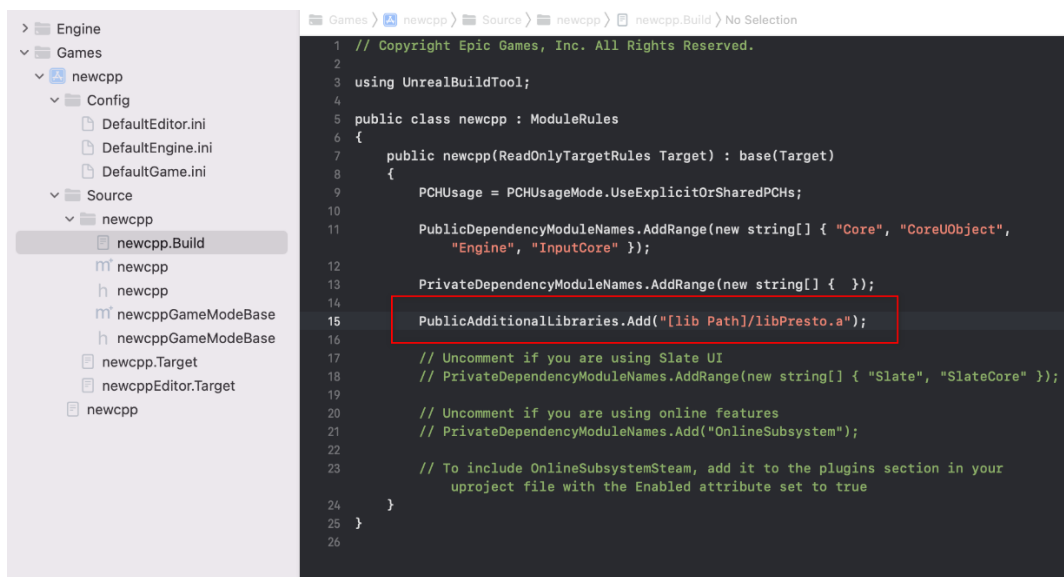
**보안 적용이 되지 않은 경우에는 스피드해킹, 메모리변조가 감지되면 앱이 비정상 종료될 수 있습니다.**

# G-PRESTO Unreal Engine 프로젝트 적용

## UNREAL ENGINE 프로젝트 적용

### 1. 라이브러리 등록

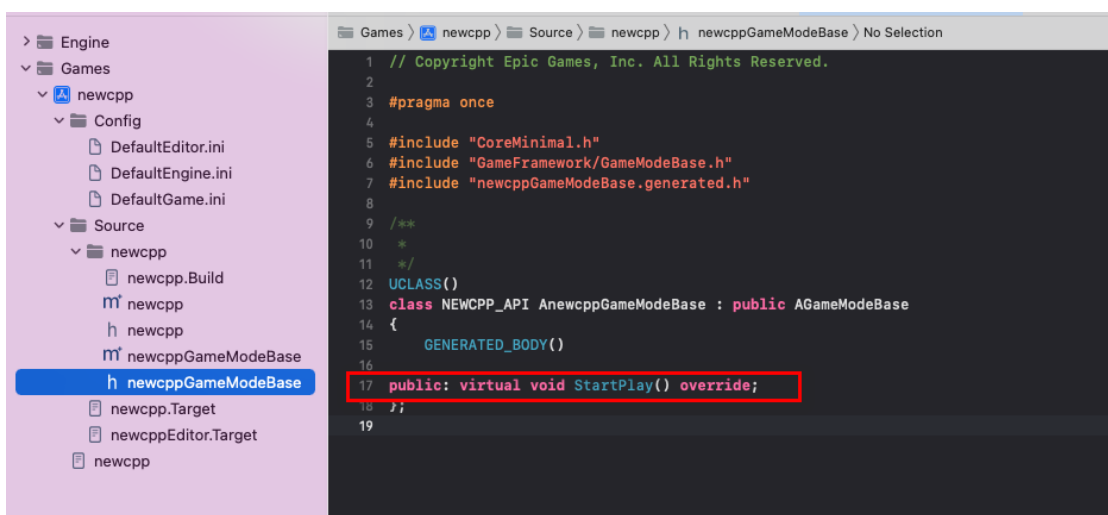
프로젝트이름.Build 에 PublicAdditionalLibraries.Add("[lib Path]/libPresto.a"); 추가 합니다.



<라이브러리 경로 추가>

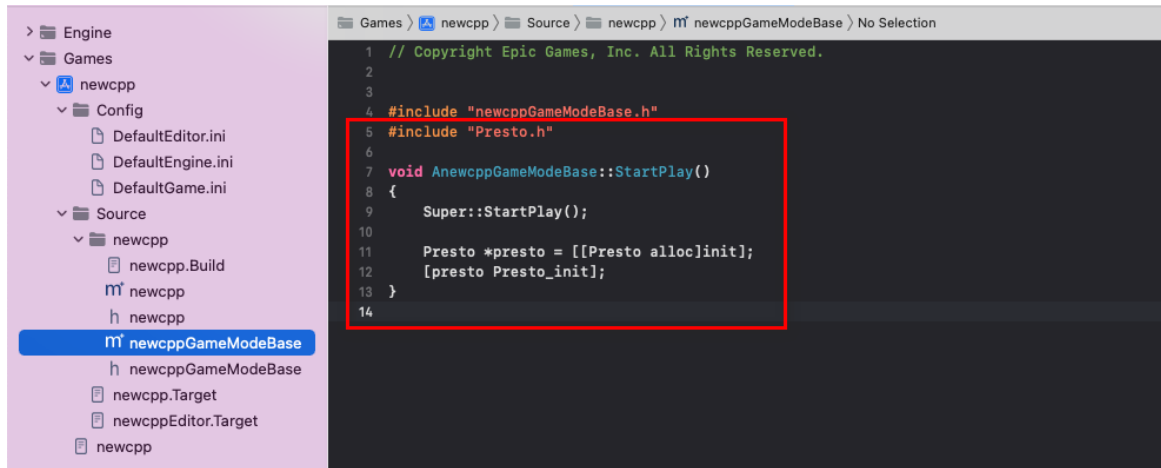
### 2. [프로젝트이름]GameModeBase 에 시작 코드 추가

게임 실행 초기에 실행 코드가 실행되도록 GameModeBase 가 아니라 초기에 실행되는 코드 원하는 곳에 엔진 실행 코드를 추가합니다.



<예시로 StartPlay() 실행을 위해 헤더에 추가>

## G-PRESTO Unreal Engine 프로젝트 적용

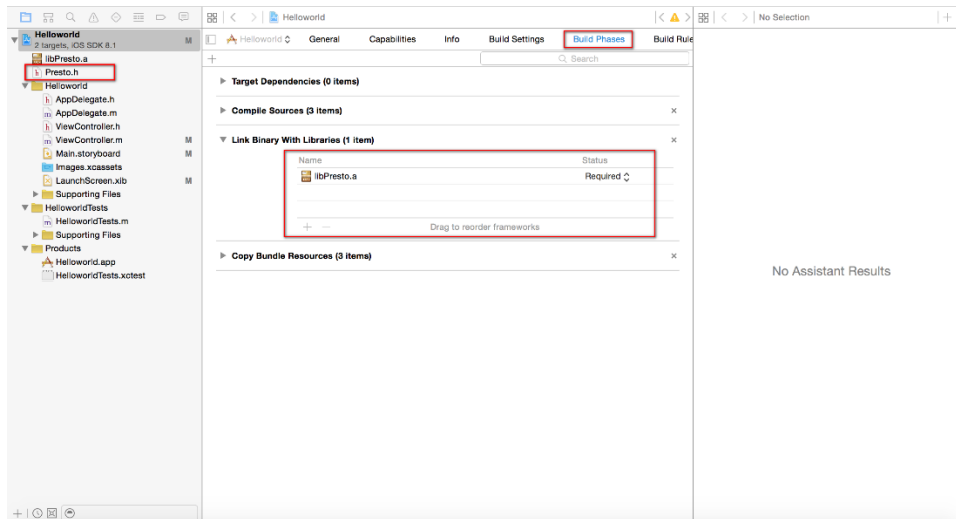


<예시로 StartPlay()에 엔진 실행 코드를 추가>

## G-PRESTO 그외 프로젝트 적용

### G-PRESTO XCODE 에서 적용

**참고** : Unity 프로젝트의 경우 아래의 Xcode 에서 적용하는 작업을 진행하지 않습니다.



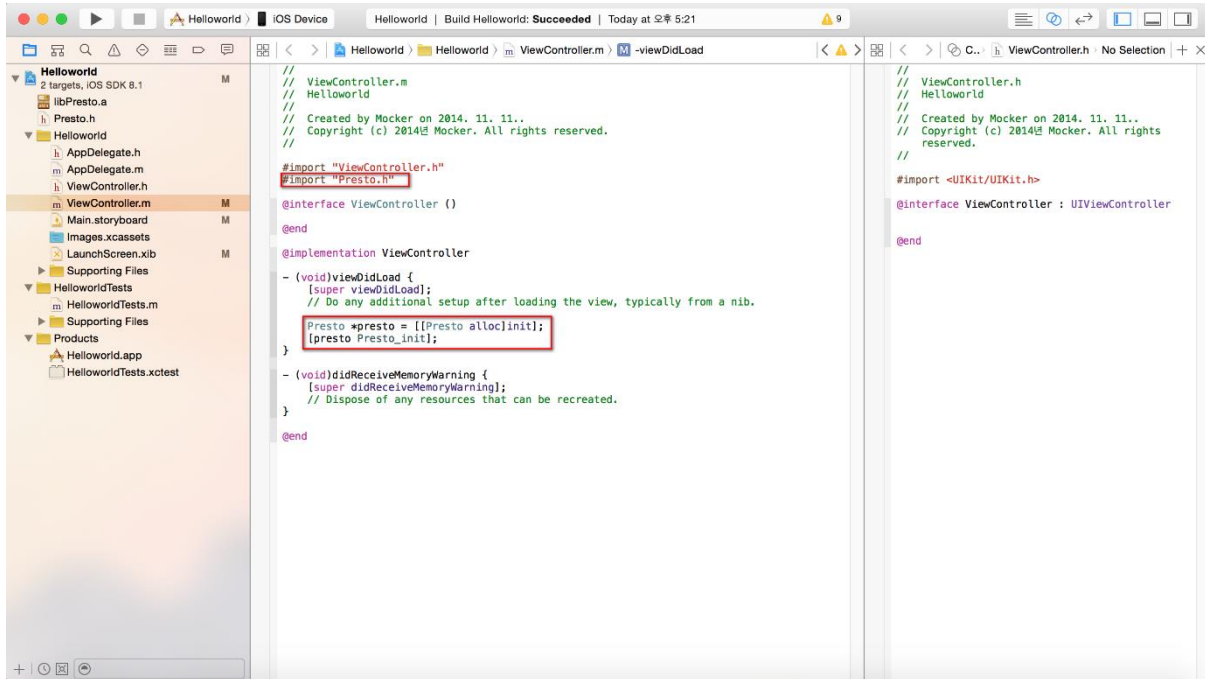
<프로젝트에 Presto 라이브러리 추가>

### G-PRESTO 라이브러리 추가

1. 프로젝트에 Build Phases 메뉴 클릭
2. Link Binary With Libraries에 libPresto.a 파일을 드래그하여 삽입
3. 프로젝트에 Presto.h 파일을 드래그하여 삽입
4. Build Settings > Search Paths > Library Search Paths에 라이브러리 경로 추가

## G-PRESTO 그외 프로젝트 적용

### G-PRESTO IOS 실행 코드 적용 [OBJECTIVE-C]

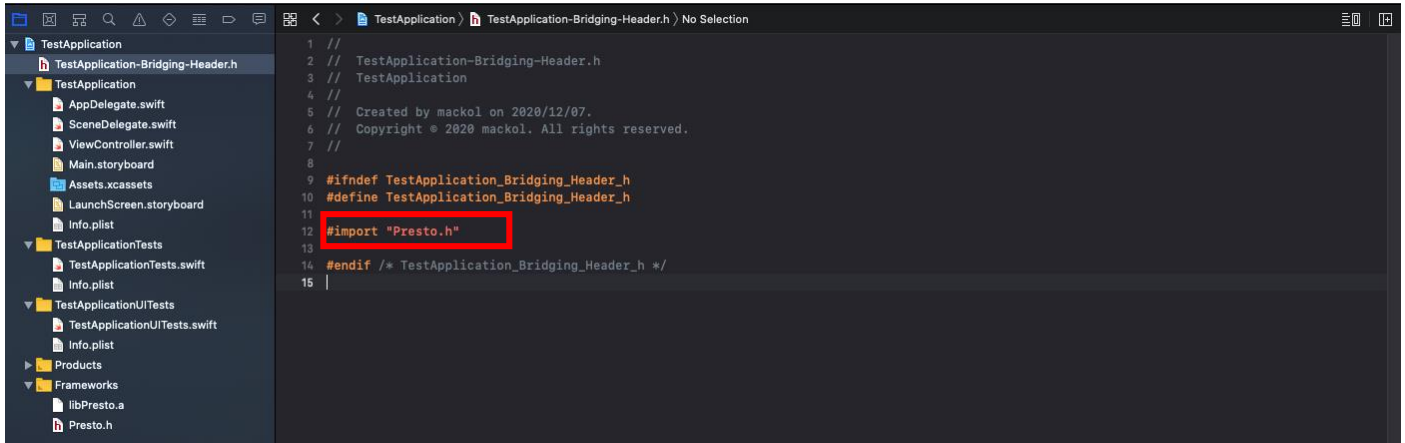


<Objective-C Presto 실행 코드 삽입>

원하시는 시점에 Presto 실행 코드를 입력하고 #import "Presto.h"를 작성하여 헤더 파일을 찾을 수 있도록 작성하면 됩니다. 그리고 Presto 라이브러리를 호출할 수 있도록 화면과 같이 코드를 작성하면 됩니다.

## G-PRESTO 그외 프로젝트 적용

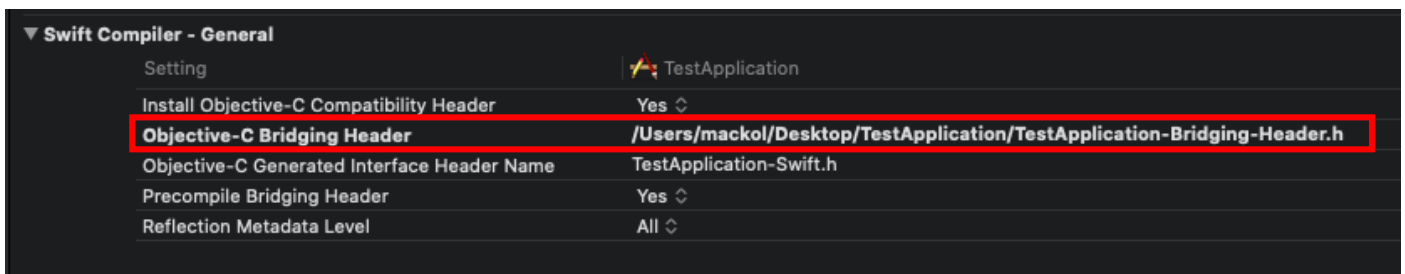
### G-PRESTO IOS 실행 코드 적용 [SWIFT]



```

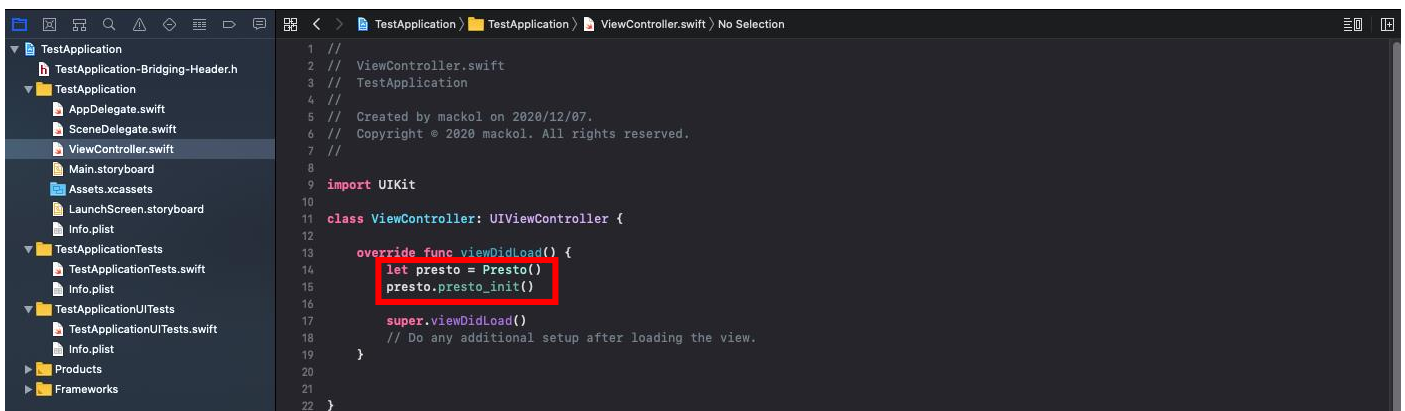
1 //
2 // TestApplication-Bridging-Header.h
3 // TestApplication
4 //
5 // Created by mackol on 2020/12/07.
6 // Copyright © 2020 mackol. All rights reserved.
7 //
8
9 #ifndef TestApplication_Bridging_Header_h
10 #define TestApplication_Bridging_Header_h
11
12 #import "Presto.h"
13
14 #endif /* TestApplication_Bridging_Header_h */
15

```



Swift Compiler - General	
Setting	TestApplication
Install Objective-C Compatibility Header	Yes
Objective-C Bridging Header	/Users/mackol/Desktop/TestApplication/TestApplication-Bridging-Header.h
Objective-C Generated Interface Header Name	TestApplication-Swift.h
Precompile Bridging Header	Yes
Reflection Metadata Level	All

<Swift bridging-header 내부에 Presto.h 추가>



```

1 //
2 // ViewController.swift
3 // TestApplication
4 //
5 // Created by mackol on 2020/12/07.
6 // Copyright © 2020 mackol. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         let presto = Presto()
15         presto.presto_init()
16
17         super.viewDidLoad()
18         // Do any additional setup after loading the view.
19     }
20
21 }
22

```

<Swift Presto 실행 코드 삽입>

G-Presto 라이브러리를 추가 후 Swift에서 Objective-C 객체를 참조하기 위해 bridging-header.h 생성하거나, 기존에 사용 중인 헤더가 있다면 내부에 #import "Presto.h"를 작성하여 헤더 파일을 찾을 수 있도록 작성하면 됩니다. 그리고 원하는 시점에 실행 코드를 입력하여 Presto 라이브러리를 호출할 수 있도록 화면과 같이 코드를 작성하면 됩니다.

## G-PRESTO 엔진 난독화 Bitcode 설정

### 엔진 코드 BITCODE 설정

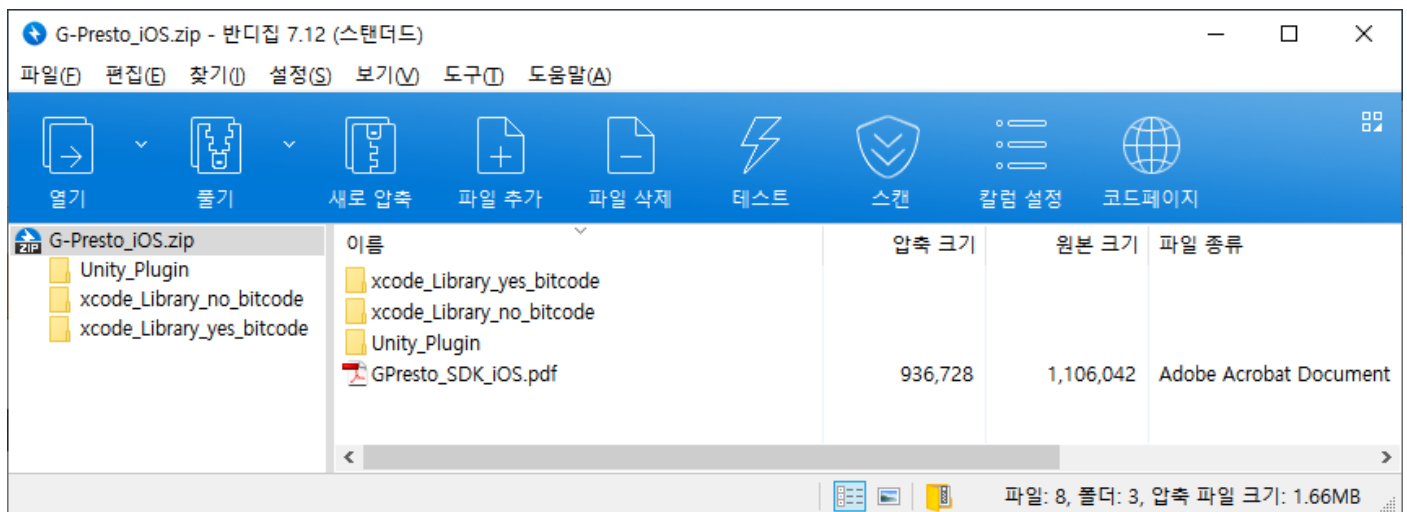
G-Presto 엔진은 보안 기능이 적용된 LLVM 컴파일러를 사용하여 bitcode 가 적용되지 않습니다.

Xcode Build Options -> Enable Bitcode를 NO 로 설정합니다.

Bitcode가 적용될 경우 Archive 시 오류가 발생합니다. ([참조페이지-문제해결](#))

```
ld: bitcode bundle could not be generated because '/Users/사용자/Desktop/il2cppTest/ios/Libraries/Plugins/iOS/libPresto.a(Presto.o)' was built without full bitcode. All object files and libraries for bitcode must be generated from Xcode Archive or Install build for architecture arm64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

< Bitcode 적용으로 인한 오류 >



프로젝트에 따라 용량 등 기타 문제를 극복하기 위해 Bitcode 사용이 불가피 한 경우를 위해 G-Presto iOS SDK 에 Bitcode가 적용된 버전(xcode\_Library\_yes\_bitcode)과 적용되지 않은 버전(xcode\_Library\_no\_bitcode)를 제공하고 있습니다.

Bitcode가 적용된 버전은 LLVM 컴파일러를 사용하지 않기 때문에 엔진 난독화 기능이 적용되지 않습니다.

Bitcode가 적용되지 않은 버전은 엔진 난독화 기능이 적용된 버전입니다.

특수한 경우를 제외하고, 프로젝트의 보안성 향상을 위해 난독화 기능이 제공된 xcode\_Library\_no\_bitcode를 사용 하시기 바랍니다.

## G-PRESTO UUID 생성

### UUID 를 통한 치트 유저 확인

단말기에 통신 칩이 없는 Wi-Fi 전용 단말기의 경우 DeviceId 값이 없기 때문에 치트 유저를 추적하기가 어렵지만, UUID를 서버에 기록해 G-Presto서비스 사이트에서 매칭하면 원하는 유저를 파악이 가능함.

Xcode

```
NSString *uuid = [NSString string];
uuid = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
uuid = [uuid stringByReplacingOccurrencesOfString:@"-"
                                     withString:@""];
```

Unity

```
//G-Presto UUID( iOS)
//추후 G-Presto 서버와 매칭을 위해 유저의 정보를 게임 서버로 전송 시 해당 함수를 호출하여 값을 게임 서버로 전송
string uuid = GPresto.Protector.Engine.GPrestoIOS.GPiOS_UUID();
Debug.Log("G-Presto UUID = " + uuid);
```



## G-PRESTO 실시간 검사 옵션 적용

### 실시간 검사 옵션 적용

패키지 설정

게임명

Search:

	치트툴차단	앱위변조차단	가상머신차단	매크로차단	dex난독화	Res암호화	so난독화	unity암호화	루팅차단
	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF
igin.google	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF	<input type="checkbox"/> OFF

iOS 버전에서는  
사용되지 않는 기능

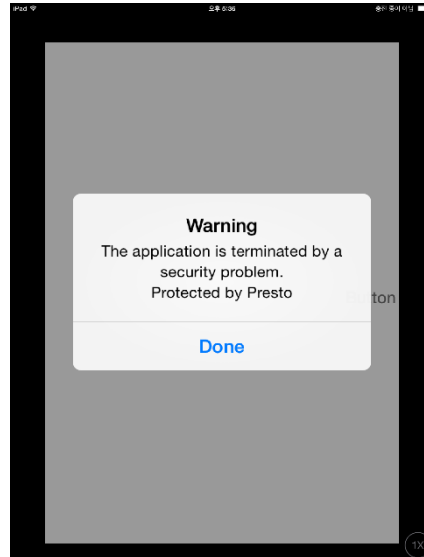
#### <실시간 검사 옵션>

안드로이드 운영체제와는 달리 iOS 운영체제의 특성상 탈옥 차단, 치트툴 차단, 앱위변조차단 기능이 동작하며 실시간으로 옵션을 변경할 수 있습니다.

**참고** : iOS는 치트툴 차단 옵션을 OFF 시 로그가 남지 않습니다. 따로 요청 시 로그가 남도록 변경 가능합니다.

## G-PRESTO 테스트

### 탈옥 단말기 테스트



아래의 버전별 탈옥툴 주소를 참고하여 단말기를 탈옥하고 Presto iOS가 적용된 앱을 실행하면 상기 화면과 같이 경고 문구가 화면에 출력되며 버튼 클릭시 앱이 정상 종료되는지 테스트

콘솔 로그를 통해 확인할 수 있는 툴

iMazing

<https://imazing.com/>

iOS 11.0 – 13.5 버전 탈옥

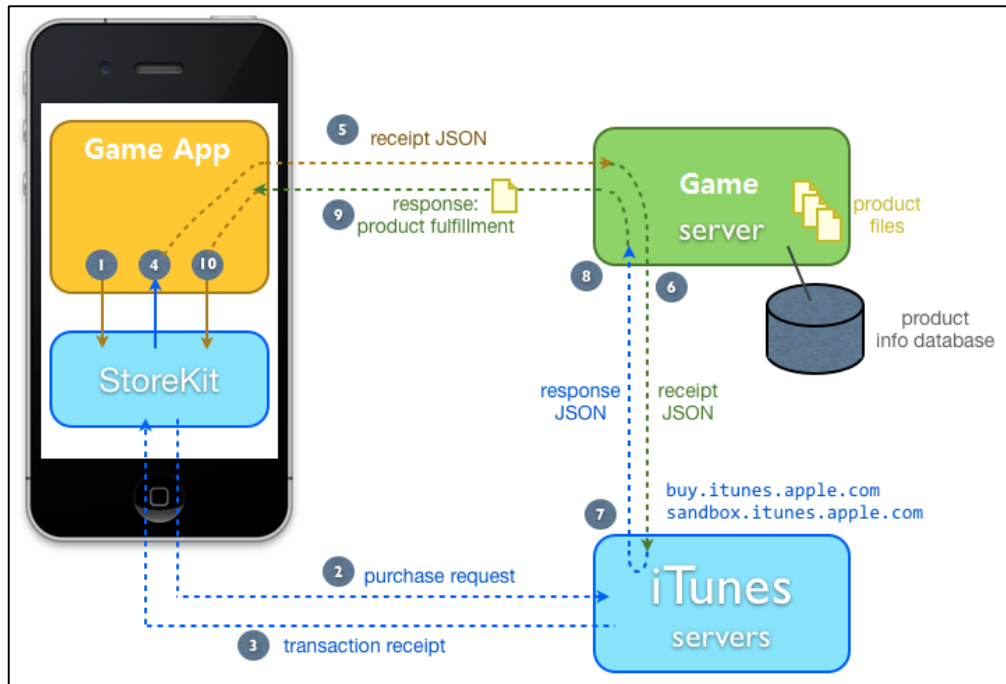
<https://unc0ver.dev/>

※ xcode에서 실행시 참고 사항

G-Presto 연동한 상태에서 실제 디바이스 연결 테스트시 앱이 경고 없이 종료된다면 Xcode 상에서 실행하지 마시고 실제 디바이스에 올린 파일을 클릭하여 실행하시기 바랍니다. G-Presto 내에 디버깅 방지 코드가 삽입되어 디버거가 감지되면 종료됩니다. 기능을 중지하시려면 G-Presto 초기화 코드를 삭제 주석 처리 후 실행하시기 바랍니다. (Unity 예시) // `GPresto.Protector.Engine.GPrestoEngine.Start();`

## 인앱 결제

### 인앱 결제 영수증 검증



<인앱 결제 검증 흐름>

### 인앱 결제 확인 사항

1. 인앱 결제에 대한 검증을 앱 내에서 진행하면 안되며, 반드시 서버에서 검증
2. 애플 서버를 통해 검증에 성공한 결과에 대해 영수증 데이터는 추후 문제 발생시 사용될 수 있으므로 서버 로그에 보관
3. 'transaction\_id'는 동일한 경우가 없기 때문에 결재한 영수증을 여러 번 보내서 인앱 결제를 우회하는 경우 'transaction\_id'의 검증을 통해 정상 구매 여부를 확인
4. 대부분의 인앱 우회 툴의 경우 임의의 상품명으로 구매 결과를 보내기 때문에 'product\_id', item\_id" 등의 상품명 관련 정보의 검증을 통해 구매 여부를 확인
5. 'purchase\_date' 구매 날짜를 검증하여 구매 여부를 확인

### 인앱 결제 영수증 검증 구현 참고 사이트

<http://blog.miyu.pe.kr/298>

## 크로스체크(옵션기능)

### 크로스체크 UNITY 에서 함수 호출

```
string sData = GPresto.Protector.Engine.GPrestoEngine.GetData();  
Debug.Log("G-Presto sData = " + sData);
```

상기 호출 예시를 참고하여 Unity에서 GetSessionData() 함수를 호출하여 sData를 전달받아 안드로이드와 동일하게 게임 서버로 해당 값을 전달하여 해당 유저가 정상적인 유저인지를 확인하고 로그인을 진행 하거나 차단 합니다.(서버 작업이 필요하며 서버 소스를 당사 담당자에게 요청하십시오.)

### 크로스체크 UNITY 이외 함수 호출

```
NSString *sData = [presto Presto_getsDataNS];
```

### 주의 사항

1. 크로스체크를 위해 앱의 패키지명이 저희 서버에 등록되어야 합니다. 요청 주시기 바랍니다.
2. 로그인 과정에서 해당 Presto\_getsDataNS() 함수를 호출하도록 합니다. (로그아웃 이후 다시 로그인 할 때도 다시 호출되어야 함)
3. 앱이 업데이트 되는 경우 반드시 앱의 버전 넘버를 변경해야 합니다. (버전 넘버에 따라 고유값이 서버에 저장되기 때문에 같은 버전 넘버를 사용하면 안됨)
4. 앱이 업데이트 이후 G-Presto 서버가 앱의 고유값을 충분히 축적한 뒤에 비정상 값을 파악하여 차단하는 방식으로 동작합니다.

## 자주 묻는 질문 (FAQ)

### 자주 묻는 질문 FAQ

#### 1. iOS 지원 환경은 어떻게 되나요?

- iOS 6.x 버전 이상 지원합니다.
- Xcode 7.x 버전 이상 지원합니다.

#### 2. iOS 버전은 난독화, 암호화 기능이 없나요?

- iOS 빌드에 사용되는 LLVM(Low Level Virtual Machine) 컴파일러에 난독화 기능을 추가하여 아래와 같은 기능이 G-Presto 엔진에 적용됩니다.
- Bogus Control Flow, Control Flow, Flattening Instruction Substitution, String Obfuscation
- Unity plug-in 제공하는 변수 암호화 기능을 통해 메모리 변조를 차단할 수 있습니다.

#### 3. G-Presto iOS 버전의 탐지 방식에 대해서 궁금합니다.

- iOS 는 9.x 버전부터 프로세스 목록을 구할 수 없어 프로세스로 실행중인 목록을 탐지 할 수 없습니다.
- 설치 경로를 통해 검색을 하고 있으며, iOS 의 폐쇄적인 권한으로 인하여 검색에 한계가 있습니다.
- 메모리 해킹 툴이나 매크로 등 다양한 툴을 차단하고 있으나, 말씀드린 사항처럼 한계가 있기 때문에 인앱 결제 우회 툴의 경우 IAP 우회 테스트 및 영수증 검증을 필히 적용하실 수 있도록 가이드 하고 있습니다.

#### 4. G-Presto iOS 의 앱위변조 탐지 방식이 궁금합니다.

- 클라이언트에서 위변조검사를 통해 앱 위변조를 탐지하고 있습니다.
- 클라이언트 내부 위변조검사는 비교적 쉽게 우회 될 수 있어서 크로스체크를 적용하도록 하고 있습니다.
- 크로스 체크를 통해 게임 서버와 G-Presto 서버를 연계하여 앱의 무결성 검증을 진행하며 앱의 보안성을 강화하게 됩니다.

## 자주 묻는 질문 (FAQ)

### 문제 해결

- G-Presto 엔진 Bitcode 미적용으로 인해 빌드 오류

```
ld: bitcode bundle could not be generated because '/Users/사용자/Desktop/il2cppTest/ios/Libraries/Plugins/iOS/libPresto.a(Presto.o)' was built without full bitcode. All object files and libraries for bitcode must be generated from Xcode Archive or Install build for architecture arm64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

G-Presto 엔진은 보안 기능이 적용된 LLVM 컴파일러를 사용하여 엔진 빌드시 bitcode 가 적용되지 않습니다.

프로젝트 빌드시 Build Options -> Enable Bitcode 를 NO 로 설정하여 프로젝트에서 Bitcode 적용하지 않도록 설정하면 Unity 2018 까지는 문제가 없었지만 이후 버전에서는 제대로 적용되지 않는 문제가 있습니다.

유니티에서 빌드로 생성된 xcode 프로젝트 내부에 Unity-iPhone.xcodeproj 파일의 패키지 내용 보기를 해보시면 project.pbxproj 파일이 있습니다. **ENABLE\_BITCODE** 으로 검색하면 **/\*Debug \*/ /\* Release \*/** 으로 시작되는 설정값을 찾을 수 있습니다. 두 곳의 **ENABLE\_BITCODE** 를 NO 로 변경하시면 됩니다.

```
C01FCF4F08A954540054247B /* Debug */ = {
    isa = XCBuildConfiguration;
    buildSettings = {
        ARCHS = armv7;
        "CODE_SIGN_IDENTITY[sdk=iphoneos*]" = "iPhone Developer";
        DEBUG_INFORMATION_FORMAT = dwarf;
        ENABLE_BITCODE = YES;
        ...
    };
    name = Debug;
};
```

<Unity-iPhone.xcodeproj 내부 project.pbxproj 일부>