

G-PRESTO 사용 가이드

윈도우 연동 기본 가이드

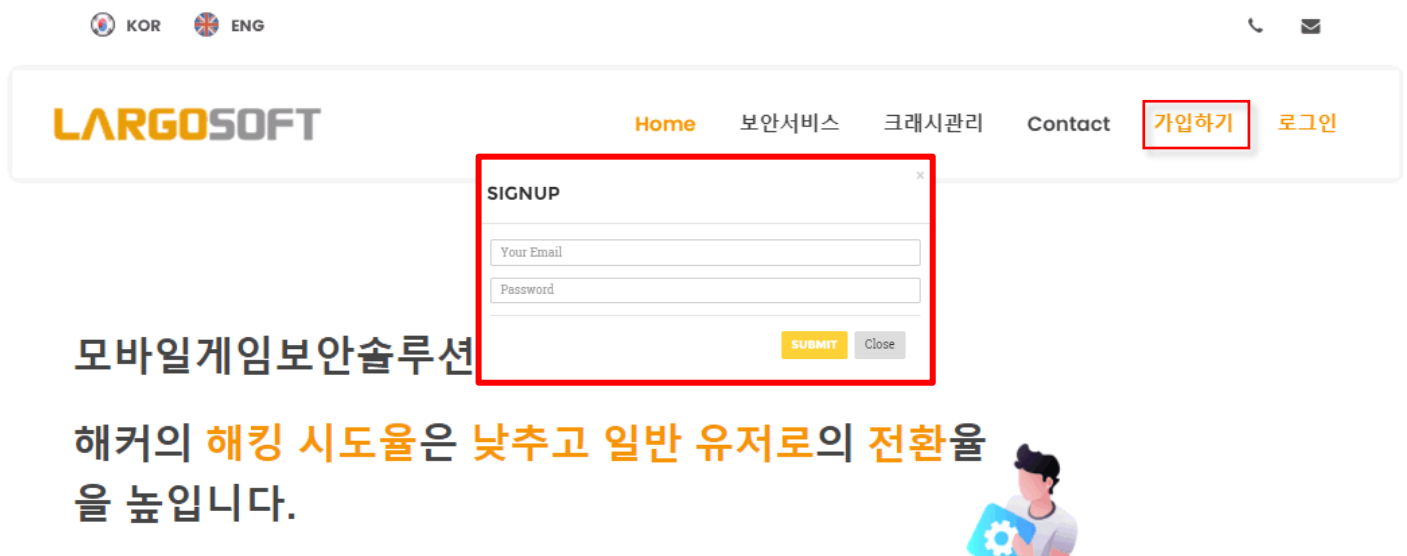
목차

회원가입 및 로그인	3
추가정보 입력	4
패키지 설정(게임 정보 등록)	5
사이트 보안 적용 메뉴 설명	6
UNITY 지원환경	7
UNITY 플러그인	7
UNITY 플러그인 프로젝트 설정	7
G-Presto 변수 사용 예제	9
스피드핵 감지 사용 예제	11
G-Presto Unity Engine 관련 예제	12
Unity 메타 데이터 암호화	13
G-Presto Engine Code 설명	16
사용자 정보 확인	18
G-PRESTO Log Data 참고사항	19
G-PRESTO Log Data JSON 응답 코드 형식	19
G-PRESTO Log Data 요청	19
G-PRESTO Blacklist 요청	20
G-PRESTO 해킹커뮤니티 배포 앱 사용자 Data 요청	21
G-PRESTO 해킹커뮤니티 배포 앱 사용자 Data 요청 참고사항	21
빌드툴 통합	22

G-PRESTO 보안 적용 사전작업

회원가입 및 로그인

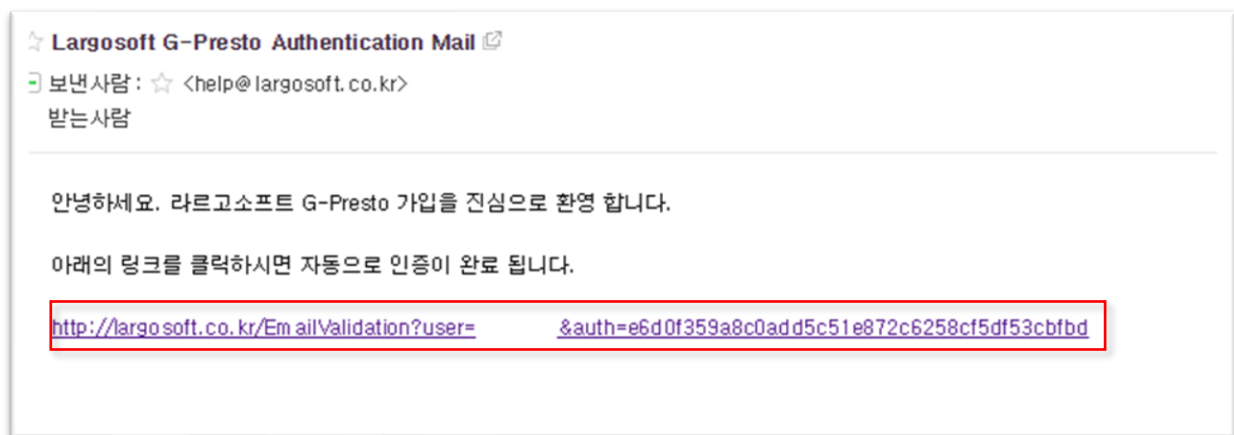
1. LARGOSOFT 사이트(<https://console.largosoft.co.kr>) 방문 후 가입하기



모바일게임보안솔루션

해커의 해킹 시도율은 낮추고 일반 유저로의 전환율을 높입니다.

2. 인증메일 확인



☆ 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>

3. LARGOSOFT 사이트 로그인 -> G-Presto 사이트로 자동이동

4. 향후 G-Presto 사이트로 직접 로그인

G-PRESTO 보안 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 가능합니다.

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	앱워변조차단	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. 게임명 등록 – 개발사에서 향후 통계분석 툴 확인 시 프로젝트를 구분 짓기 위한 분류명.

2. 패키지명 등록 – 게임명과 매칭되어 해킹통계구분에 사용됨.

- PC 버전은 패키지명을 등록할 때에 보안 적용할 어플리케이션 이름과 동일하게 등록하셔야 합니다.

Ex) 보안을 적용할 어플리케이션의 파일 이름이 SampleGame.exe 라면 SampleGame 으로 등록하셔야 합니다.

해당 내용까지 완료 시 사전작업 완료

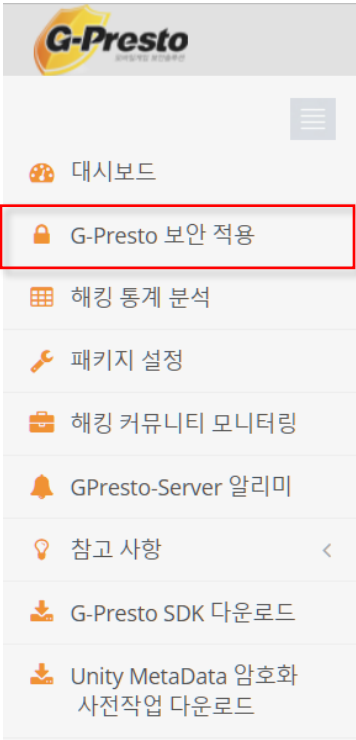
3. 하단의 패키지 설정에서 치트돌차단 옵션을 활성화.

- 치트돌차단 옵션을 활성화를 하지 않는다면 위협을 탐지해도 프로세스를 종료하지 않습니다.
- 앱이 구동되기 전 Themida 에서 위협을 탐지하면 옵션 여부와 상관없이 프로세스를 종료합니다.
- 치트돌차단 옵션은 실시간으로 반영됩니다.

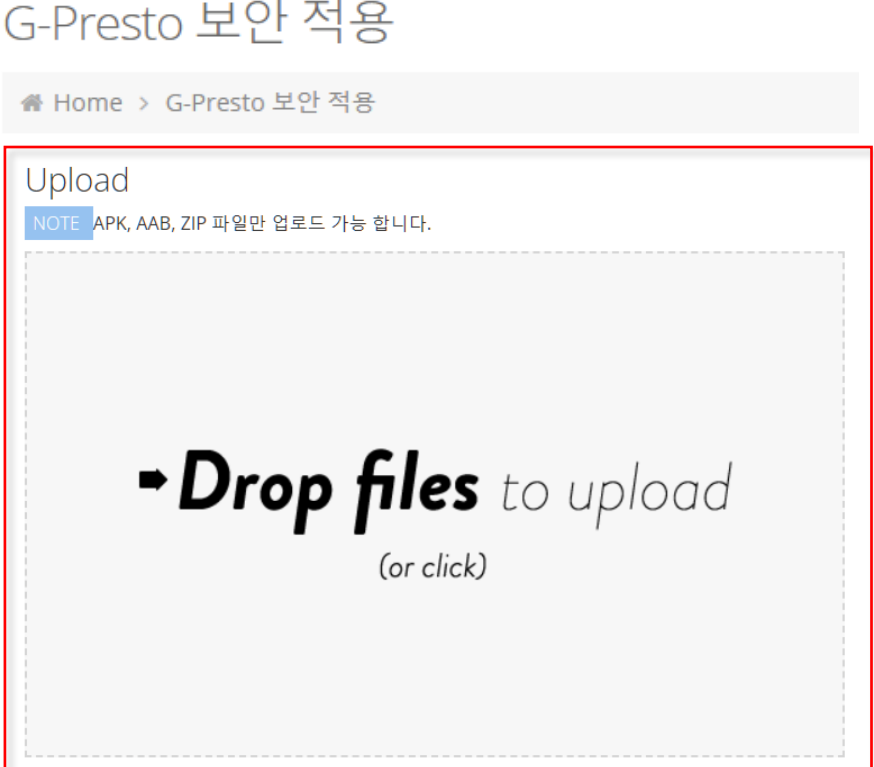
G-PRESTO 보안 적용

사이트 보안 적용 메뉴 설명

1



2



1. G-Presto 보안 적용 메뉴 선택.
2. 보안 적용을 진행할 파일 **업로드**.
3. 보안 적용 완료 후 다운로드 자동 진행.

G-PRESTO Unity 플러그인 적용

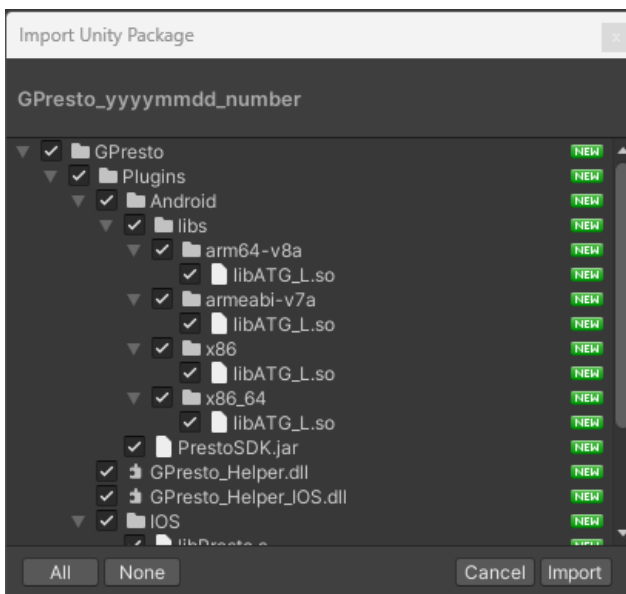
UNITY 지원환경

Unity 3.x 이상 환경에서 지원합니다.

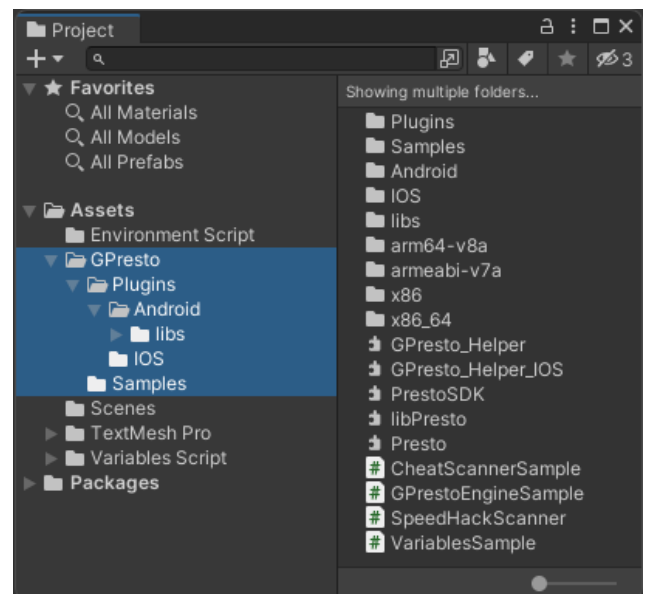
UNITY 플러그인

G-Presto 는 치트툴에 의한 메모리 위변조를 방지하기 위해 Unity 용 변수 암호화 플러그인과 G-Presto 보안 모듈을 호출하는 플러그인을 제공합니다.

UNITY 플러그인 프로젝트 설정



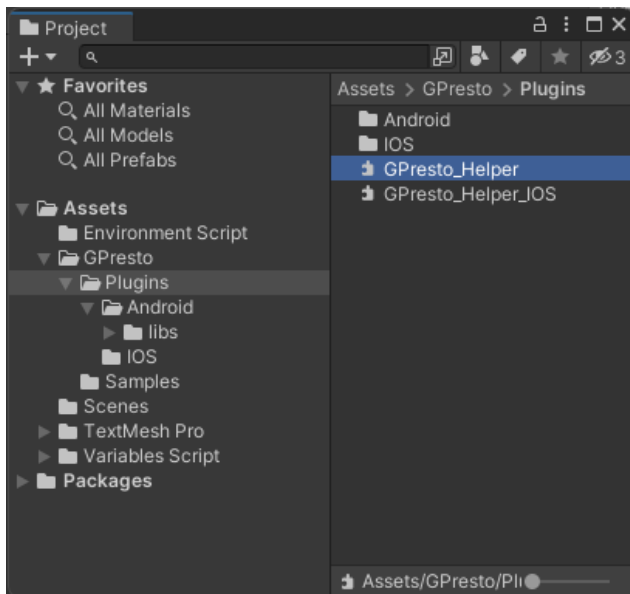
<Unity 프로젝트 import 창>



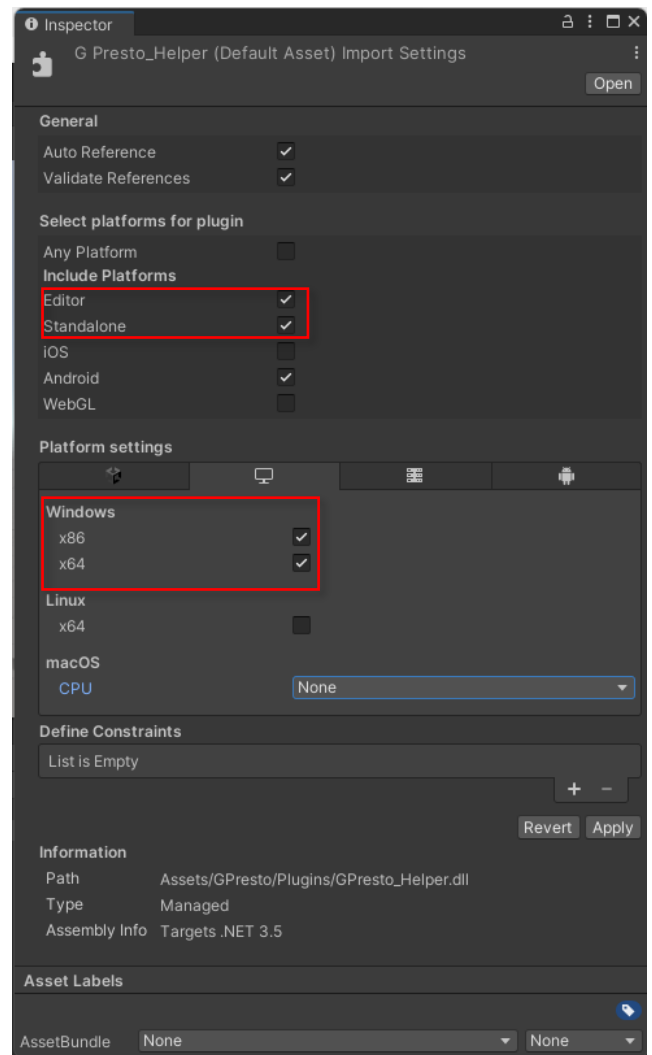
<Unity 프로젝트 import 결과>

1. Assets -> Import Package -> Custom Package
2. 전달받은 GPresto_yyyymmdd_number.unitypackage 파일 선택
3. '<Unity 프로젝트 import 창>' 오른쪽 하단에 보이는 것처럼 import 버튼을 클릭
4. '<Unity 프로젝트 import 결과>'와 같이 파일이 생성되는지 확인

G-PRESTO Unity 플러그인 적용



<Unity 프로젝트 창>



<G-Presto_Helper Inspector 창>

5. '<Unity 프로젝트 창>'에 보이는 것 같이 GPresto -> Plugins -> GPresto_Helper.dll 을 클릭
 6. GPresto_Helper.dll 의 Inspector 에서 '<G-Presto_Helper Inspector 창>'처럼 Include Platforms 에 존재하는 Editor 와 Standalone 을 선택
 7. Standalone 선택 후 생성되는 Platform settings(모니터 아이콘)에서 빌드하고자 하는 Windows 의 플랫폼을 선택
- GPresto_Helper.dll 은 Windows 의 x86, x64 를 모두 선택하셔도 됩니다.

주의: 이미지에 존재하는 파일명과 실제로 제공되는 파일명이 다를 수 있습니다.

G-PRESTO Unity 플러그인 적용

G-PRESTO 변수 사용 예제

* G-Presto 변수 내부에는 보안 로직이 존재하기 때문에 중요 변수들만 사용하는 것을 권장합니다.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using GPresto.Protector.Variables;

참조 0개
public class scoreScript : MonoBehaviour {

    public GPInt testInt = 0;
    private GPVector3 testVector3_1 = Vector3.zero;
    private GPVector3 testVector3_2 = Vector3.zero;
    private GPFLOAT testFloat = 0.0f;
}
```

부분1

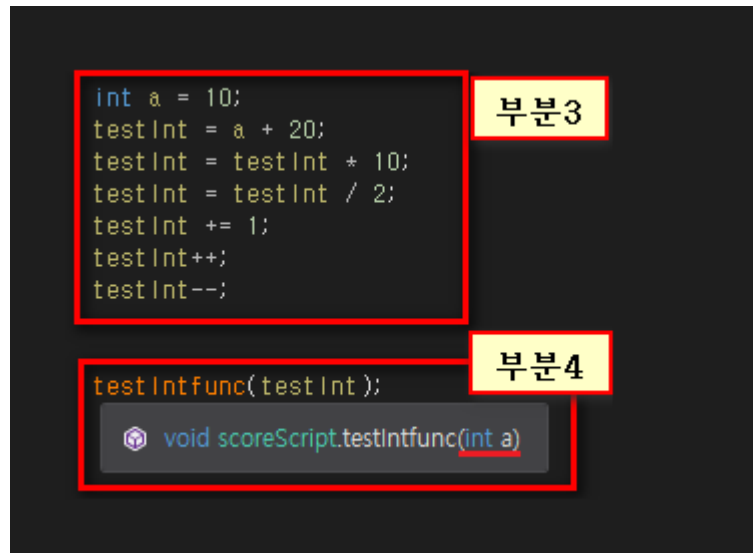
부분2

- GPBool
- GPByte
- GPChar
- GPDecimal
- GPDouble
- GPFloat
- GPInt
- GPLong
- GPQuaternion
- GPSByte
- GPShort
- GPString
- GPUInt
- GPULong
- GPUShort
- GPVector2
- GPVector3

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

1. G-Presto 자료형을 사용하기 위하여 네임스페이스 추가 (<부분 1> 참고)
2. G-Presto 자료형 선언 예시 (<부분 2> 참고)
(일반적인 자료형 선언과 동일하게 사용 가능)

G-PRESTO Unity 플러그인 적용



<Unity C#Script GPInt 사용 예>

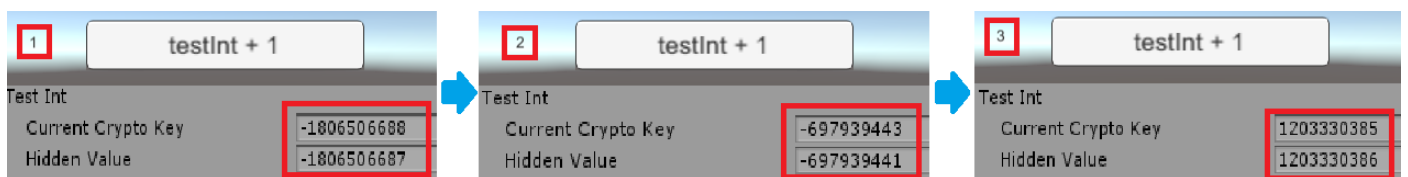
3. '<부분 2>'에 선언된 GPInt 자료형 사용 예시 (<부분 3> 참고)

(GPInt 는 int 와 같이 연산이 가능)

4. '<부분 2>'에 선언된 GPInt 자료형 사용 예시 (<부분 4> 참고)

(int 형을 매개변수로 하는 testIntfunc 함수를 GPInt 형인 testInt 변수를 전달하여 호출 가능)

*** int 뿐만 아니라 모든 GP 자료형들은 기본 자료형과 같은 방식으로 사용 가능함.**

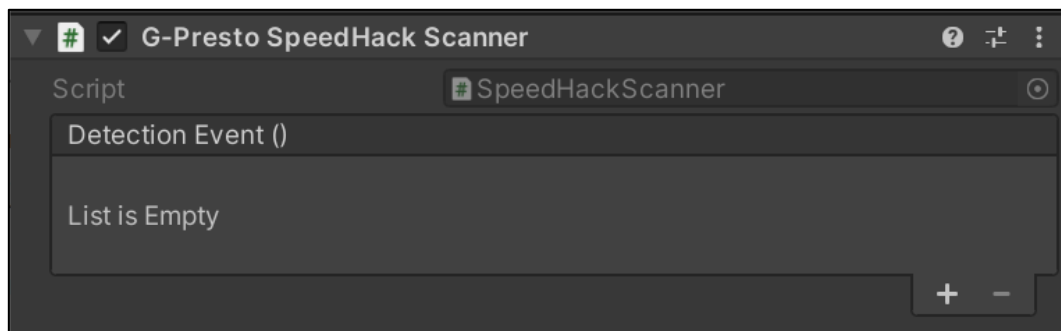


<GPInt testInt 의 암호화된 값>

G-PRESTO Unity 플러그인 적용

스피드해킹 감지 사용 예제

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



샘플로 제공되는 SpeedHackScanner.cs 를 오브젝트에 추가하시면 스피드해킹 차단 기능을 이용하실 수 있습니다.

```
private void Awake()
{
    //스피드해킹 검사 초기화
    SpeedHack.ResetStartTicks();
    Instance = this;
}

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

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

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

SpeedHack Class 를 직접 코드에 사용할시 경우 먼저 SpeedHack.ResetStartTicks() 함수를 호출하여 SpeedHack 을 초기화합니다. 초기화를 완료 후 MonoBehaviour Class 에 존재하는 Update()함수처럼 SpeedHack.IsDetect() 함수를 지속적으로 호출하게 구성하셔야 합니다.

G-PRESTO Unity 플러그인 적용

G-PRESTO UNITY ENGINE 관련 예제

G-Presto Engine 구동 여부 확인

using 문을 이용하여 GPresto.Protector.Engine 을 선언하시고 GPrestoEngine.GetStatus() 함수를 이용하시면 Engine 의 구동 여부를 확인할 수 있습니다.

GPrestoEngine.GetStatus()의 Return값이 false면 정지 상태를 나타내고 true면 동작 상태를 나타냅니다.

```
public bool GetStatus()
{
    bool result = GPrestoEngine.GetStatus();
    if (result)
    {
        Debug.Log("G-Presto Engine is running.");
    }
    else
    {
        Debug.Log("G-Presto Engine is not running.");
    }

    return result;
}
```

<Engine 상태를 얻어오는 함수 사용 예시>

Cross Platform 기반 개발

Windows 플랫폼에서 실행되는 프로그램에 보안 적용이 정상적으로 완료된 경우 보안 엔진은 프로그램 실행 시에 자동적으로 구동됩니다. 하지만 iOS 플랫폼인 경우 보안 엔진을 명시적으로 호출하셔야 엔진이 구동되기 때문에 Cross Platform 기반으로 개발하시면 GPrestoEngine.Start() 함수를 앱이 시작하는 곳에 호출하시기 바랍니다.

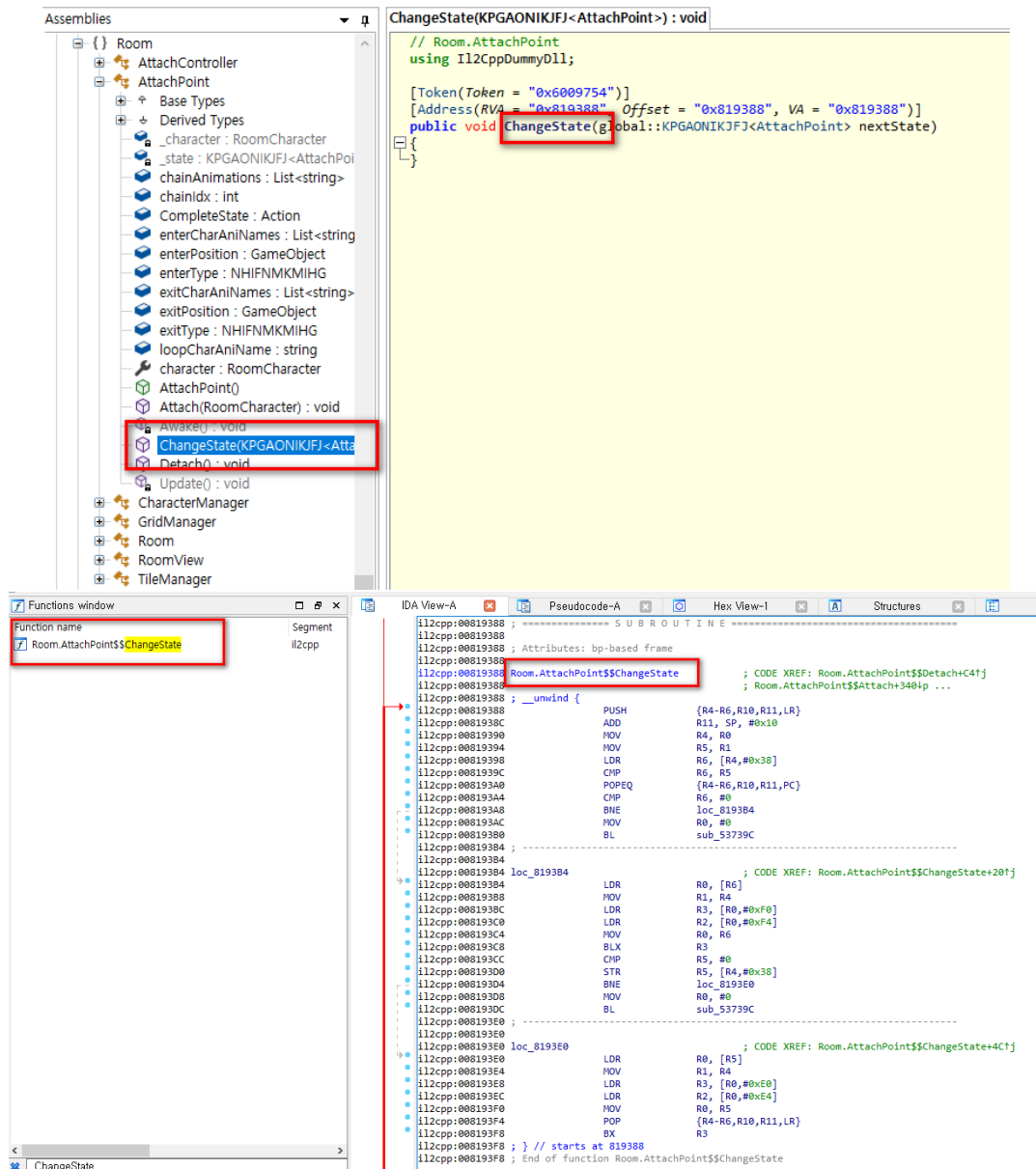
```
// 앱이 시작되는 시점에 호출
GPresto.Protector.Engine.GPrestoEngine.Start();
```

<Engine 을 구동하는 함수 사용 예시>

UNITY 메타 데이터 암호화

UNITY 메타 데이터 암호화

* 해당 기능은 IL2CPP 로 설정된 프로젝트에서만 적용이 가능합니다.



The screenshot displays the Unity development environment. On the left, the 'Assemblies' panel shows the 'Room' assembly, with 'AttachPoint' and its 'ChangeState(KPGAONIKJFJ<AttachPoint>)' method highlighted. The 'Functions window' on the bottom left shows the 'Room.AttachPoint\$\$ChangeState' function. The main window shows the C# code for the 'ChangeState' method, which uses 'Il2CppDummyDll' and 'Token' to call a native function. The bottom right shows the assembly code for this function in IDA, with the 'Room.AttachPoint\$\$ChangeState' label highlighted.

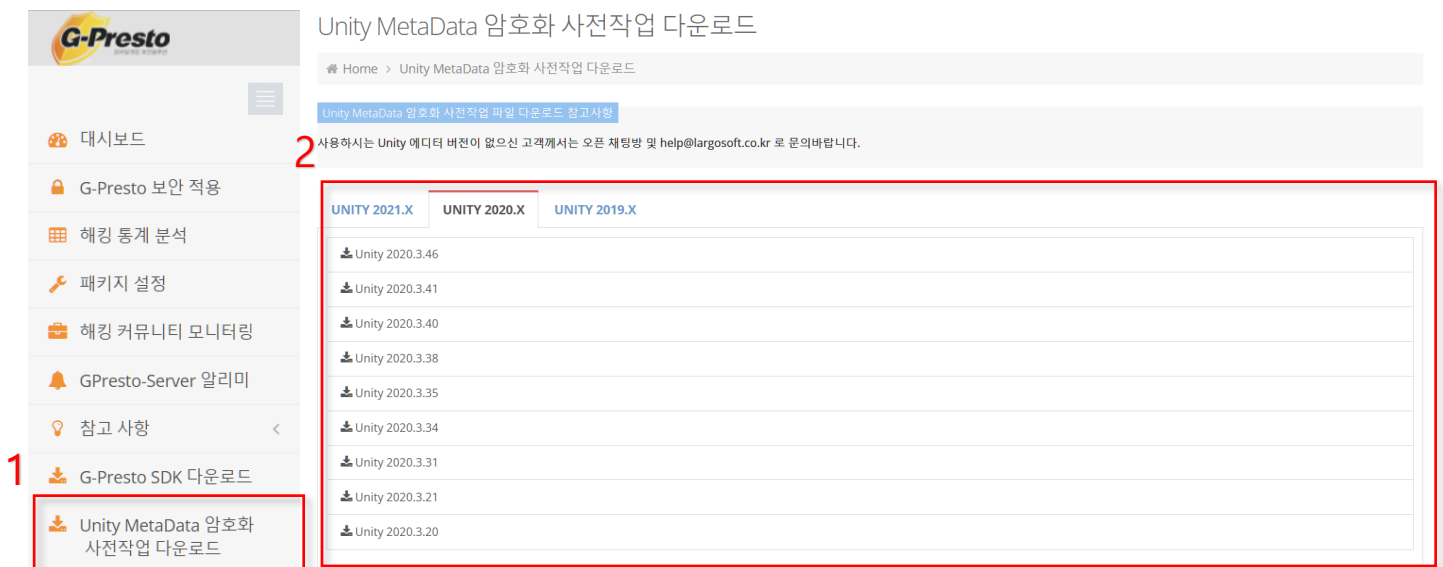
<global-metadata.dat 를 통해 매핑된 주소로 함수를 찾아 분석하는 화면>

IL2CPP 빌드시 생성되는 il2cpp.dll 을 IDA 로 분석하면 원하는 함수 주소를 찾고 분석하기가 매우 어렵지만 global-metadata.dat 파일을 이용하여 함수명과 주소를 매칭한다면 원하는 함수를 검색하고 실제 코드에 대한 정적 분석을 할 수 있습니다. 'Unity 메타 데이터 암호화' 기능은 매칭 정보가 저장된 global-metadata.dat 를 암호화하기 때문에 정적 분석을 어렵게 만들 수 있습니다.

UNITY 메타 데이터 암호화

적용 방법

1. 사용하고 계신 유니티 버전을 확인합니다.
2. largosoft 콘솔 사이트의 'Unity MetaData 암호화 사전작업 다운로드' 메뉴에서 확인하신 유니티 버전과 동일한 파일을 다운받아 아래 'OS 별 적용 파일 경로'를 참고하여 파일을 덮어씁니다.



Unity MetaData 암호화 사전작업 다운로드

Home > Unity MetaData 암호화 사전작업 다운로드

Unity MetaData 암호화 사전작업 파일 다운로드 참고사항

사용하시는 Unity 에디터 버전이 없으신 고객께서는 오른 채팅방 및 help@largosoft.co.kr 로 문의바랍니다.

UNITY 2021.X	UNITY 2020.X	UNITY 2019.X
	Unity 2020.3.46	
	Unity 2020.3.41	
	Unity 2020.3.40	
	Unity 2020.3.38	
	Unity 2020.3.35	
	Unity 2020.3.34	
	Unity 2020.3.31	
	Unity 2020.3.21	
	Unity 2020.3.20	

3. largosoft 콘솔 사이트의 '패키지 설정' 메뉴에서 Unity 암호화 옵션을 활성화합니다.

루팅가상머신차단	루팅가상화웨이브차단	ADB 차단	Dex난독화	Res암호화	Unity암호화
OFF	OFF	OFF	OFF	OFF	ON

4. 소스 코드 삽입된 빌드를 보안 적용 사이트에 업로드하여 메타 데이터 암호화가 적용된 앱 다운로드합니다.

참고

- Unity 메타 데이터 암호화 기능은 보안 향상을 위해 사용을 권장합니다.
- 복호화에 사용되는 meta_key 는 유출 등의 이유로 요청하시면 변경 가능합니다.

UNITY 메타 데이터 암호화

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

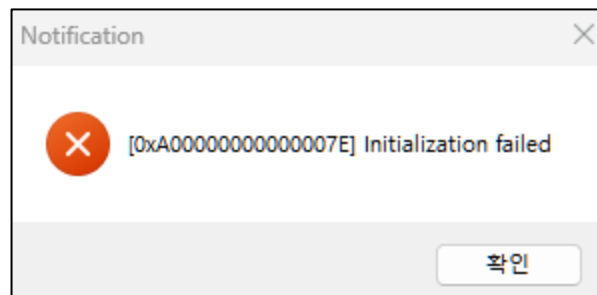
/Applications/Unity/Hub/Editor/버전/Unity.app/Contents/il2cpp/libil2cpp/vm/MetadataLoader.cpp

G-PRESTO ENGINE CODE 설명

G-PRESTO ENGINE CODE 설명

G-Presto Engine 초기화에 실패한 경우

보안이 적용된 프로세스가 실행되는 경우 G-Presto Engine이 먼저 실행되어 초기화를 진행합니다. 만약 초기화 과정을 실패한다면 '<초기화 실패 MessageBox>'처럼 MessageBox가 생성된 후 보안이 적용된 프로세스를 종료합니다.



<초기화 실패 MessageBox>

MessageBox 에서 앞에 나타나는 코드는 어느 과정에서 초기화를 실패했는지를 나타낸 결과이며 아래표는 각 코드들의 뜻을 알려주는 표입니다.

Code	뜻
0xE000'0001'0000'0000	Themida 가 적용된 어플리케이션 이름이 빈 값인 경우
0xE000'0002'0000'0000	Themida boot loader 가 실행되기 전, G-Presto Engine 구동에 실패한 경우
0xE000'0003'0000'0000	Themida boot loader 가 실행된 후, G-Presto Engine 구동에 실패한 경우
0xA000'0000'nnnn'nnnn	G-Presto Engine 를 메모리에 Load 를 할 수 없는 경우
0xB000'0000'nnnn'nnnn	G-Presto Engine 의 함수를 찾지 못하는 경우

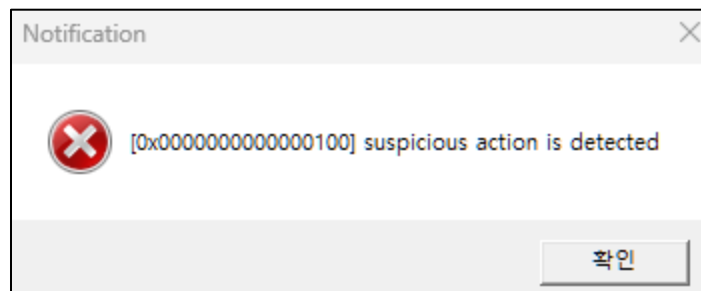
참고

- n 은 문제가 생긴 시점의 Win32 GetLastError() 함수 결과입니다.
- 0xA000'0000'nnnn'nnnn 를 나타내는 MessageBox 가 나오기 전 0xF000'0000'0000'0008 의 위협 MessageBox 를 보신다면 G-Presto Engine 의 파일이 변조되어 메모리에 Load 할 수 없는 경우입니다.

G-PRESTO ENGINE CODE 설명

G-Presto Engine에서 위협을 탐지한 경우

치트툴차단 옵션이 활성화된 상태에서 G-Presto Engine이 위협을 탐지하면 '<Engine에서 생성한 MessageBox>'처럼 MessageBox를 생성하고 G-Presto Engine을 정리 후 ExitProcess(0)함수를 호출합니다.



<Engine 에서 생성한 MessageBox>

MessageBox 에서 0x0000'0000'0000'0000 는 어떤 위협을 탐지했는지 나타낸 결과이며 아래표는 각 코드들이 어떤 뜻을 가지고 있는지 알려주는 표입니다.

Code	뜻
0x0000'0000'0000'0001	지정한 부모 프로세스가 지정한 프로세스가 아닌 경우
0x0000'0000'0000'0002	프로세스에 대한 Dump 생성을 실행한 경우
0x0000'0000'0000'0004	Debugger 를 탐지한 경우
0x0000'0000'0000'0008	앱이 변조된 것을 탐지한 경우
0x0000'0000'0000'0010	Reserved
0x0000'0000'0000'0020	가상환경에서 동작하는 경우
0x0000'0000'0000'0040	Speed Hack 을 탐지한 경우
0x0000'0000'0000'0080	Memory Cheat 를 탐지한 경우
0x0000'0000'0000'0100	Black List 의 프로세스를 탐지한 경우
0x0000'0000'0000'0200	Monitor program 을 탐지한 경우
0xF000'0000'0000'0000	Themida 를 통해서 탐지한 경우

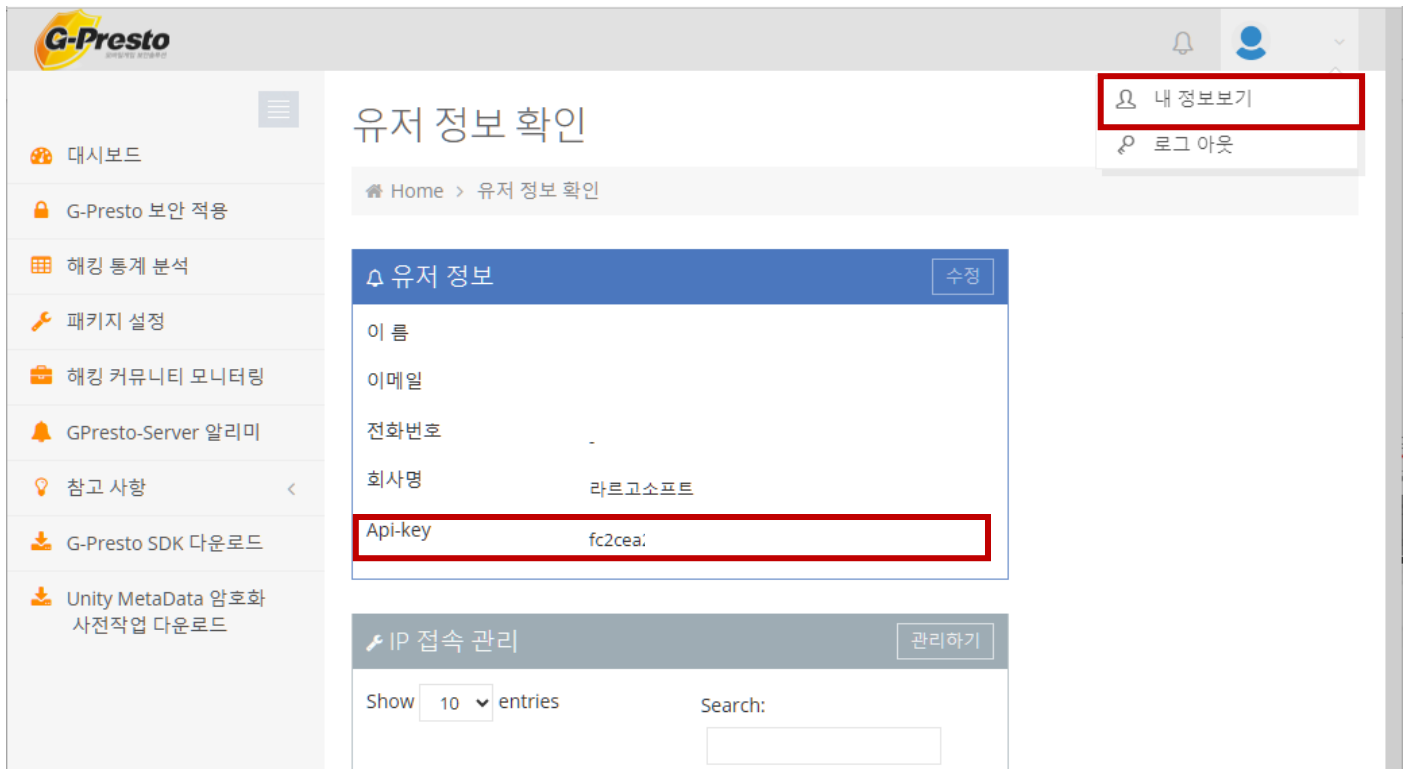
* 참고: 여러 위협이 탐지된 경우 OR 논리 연산자로 계산하여 탐지 결과가 나타납니다.

예시)

1. 0x0000'0000'0000'0082 는 Memory Cheat 위협과 Dump 위협을 탐지한 경우를 나타냅니다.
2. 0xF000'0000'0000'0004 는 Themida 에서 Debugger 를 탐지한 경우를 나타냅니다.

G-PRESTO 사용자 정보 확인

사용자 정보 확인



The screenshot shows the G-Presto user interface. On the left is a sidebar with navigation links: 대시보드, G-Presto 보안 적용, 해킹 통계 분석, 패키지 설정, 해킹 커뮤니티 모니터링, GPresto-Server 알리기, 참고 사항, G-Presto SDK 다운로드, and Unity MetaData 암호화 사전작업 다운로드. The top header features the G-Presto logo and a user profile icon. The main content area is titled '유저 정보 확인' and includes a breadcrumb 'Home > 유저 정보 확인'. Below this is a '유저 정보' section with a '수정' (Edit) button. The fields are: 이름 (Name), 이메일 (Email), 전화번호 (Phone Number), 회사명 (Company Name) - 라르고소프트, and Api-key - fc2cea:. The '로그아웃' (Logout) link in the top right is highlighted with a red box.

<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 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

```
http://largosoft.co.kr/Presto_Data?type=logs&user_id=유저아이디&user_authkey=발급받은 user_authkey&from_date=20160120&to_date=20160120 (조회할 범위 입력형식)
```

G-PRESTO Log Data

G-PRESTO BLACKLIST 요청

Blacklist 최근 7 일 조회

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

Blacklist 특정 유저의 데이터 조회

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

Blacklist 날짜 지정 조회

```
http://largosoft.co.kr/Presto_Data?type=blacklist&user_id=유저아이디&user_authkey=발급받은 user_authkey  
&from_date=20160120&to_date=20160120 (조회할 범위 입력형식)
```

Blacklist 해킹 시도 기준 설정 조회

```
http://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 사 정책과는 다를 수 있습니다. Count 를 설정하지 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

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

해킹커뮤니티 배포 앱 사용자 로그 특정 데이터 조회

```
http://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)

```
http://largosoft.co.kr/Presto_Data?type=cracklist&user_id=유저아이디&user_authkey=발급받은 user_authkey  
&from_date=20160120&to_date=20160120 (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

빌드툴 통합

Jenkins 와 같은 빌드툴에 G-Presto 보안 기능을 적용하는 프로세스를 통합하기 위해서 curl 를 이용한다면 보안 적용 페이지에 접근하지 않은 상태에서 파일을 업로드하고 보안 기능이 적용된 파일을 다운받을 수 있습니다.

주의: 파일 크기는 2GB 까지 지원합니다. 추가 확장 필요시 요청 주시기 바랍니다.

예제

- Linux

```
curl -F -file=@파일명.zip 'http://console.largosoft.co.kr/Upload?user_id=아이디&mode=a'
```

- Windows

```
curl -F -file=@파일명.zip "http://console.largosoft.co.kr/Upload?user_id=아이디&mode=a"
```

응답값

```
{
  "success":true,
  "link":"₩/WORK₩/yyyyMMddHHmmss₩/파일명.zip ",
  "fullpath":"http:₩/₩/console.largosoft.co.kr₩/WORK₩/yyyyMMddHHmmss₩/Signed_파일명.zip"
}
```

success: 성공 여부

link: 서버 주소를 포함하지 않은 서버 내에서 파일의 위치

fullpath: 서버 주소를 포함한 파일 다운 로드 경로

```
{
  "success":false,
  "errmsg":"난독화 에러 코드 : 9999, 설명 : 파일 업로드에 문제가 발생하였습니다.",
  "errorcode": "9999"
}
```

success: 성공 여부

errmsg: 실패 원인에 대한 메시지

errorcode: 실패原因的 코드

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.zip
                                "http://console.largosoft.co.kr/Upload?user_id=아이디&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>