

# SecurePlay High Score Security Component 1.0

## Overview

High scores are an integral part of games – casual and hardcore. Unfortunately, hackers and cheaters have other ideas and have sought to abuse high score systems for their advantage.

The SecurePlay High Score Security Component (HSSC) is designed to provide a low-cost, easy way to help protect against some of these hackers. The HSSC consists of a Flash client class / component and a PHP / MySQL server component. Scores are posted to the server by a simple HTTP GET request.

The HSSC works by using a challenge/response protocol to ensure the integrity and authenticity of the posted data. The HSSC cannot protect against data manipulation on the client (see our Data Obfuscator).

The HSSC is licensed on a per domain (yourdomain.com) per year basis. The goal is to provide a tool to help you protect your games at low cost.

Each game has its own cryptographic key. Make sure that the key is unique for each game title and protect the key. If you are beginning to see invalid scores, the first thing you should do is change the key for the game.

We also recommend that you use a code obfuscator, such as SWF Encrypt, to make reverse engineering of the client logic more difficult.

Please contact us at SecurePlay.com if you suspect that the system has been compromised. The HSSC is not perfect. The client Flash game is available to a determined adversary. The HSSC is designed to be easily updated while not interfering with your game site.

The HSSC is a “best effort” security tool. We do not guarantee its security and the HSSC should not be used for high value games where cash or prizes are involved. Please contact us if you are operating a high value game service. There are more advanced solutions than HSSC that may be relevant on a case-by-case basis.

## Configuring the Server

The HSSC server component includes several PHP pages and a MySQL database. All code is provided so that you can easily modify the product to fit your needs. Please be careful. Certain pages are clearly marked that they should not be modified – changing these pages may cause the HSSC to cease operating. Several sample reporting pages are provided so that you can get up and running quickly.

We have tried to use best practices to protect against malicious SQL injection and other such problems. If you find any weaknesses in our implementation, please let us know so that we can fix it for everybody.

## Creating the High Score Database

The High Score Database consists of three tables:

1. A Challenge Table that is used during the challenge/response process for posting high scores.
2. A Game Information Table that stores information about the game titles and, most critically for this application, each game title's key.
3. A High Score Table that stores the game high scores. This is a sample table and data schema. You should be able to easily extend it to meet your needs.

The current version of the HSSC does not include a user registration or account management function.

We provide a file called BuildTables.sql that will carry out a batch command from the MySQL command prompt to build the necessary tables. It is configured with the default data schema for the database. You can edit these field names, but do so with caution. Any changes need to be matched in the database configuration file (HSCDatabaseConfiguration.php).

The batch file can be executed in a couple of ways. From the shell command prompt, you can enter:

```
shell> mysql < BuildTables.sql
```

Or, if you have logged into MySQL, you can use one of the following commands:

```
mysql> source BuildTables.sql
```

```
mysql> \. BuildTables.sql
```

We do provide all of the SQL commands, so you should be able to integrate with another database easily.

## **Connecting to the Database**

The PHP code for the HSSC is designed so that you only have to modify a single file to get up and running on your server: the database configuration file (HSCDatabaseConfiguration.php).

This file contains the database connection information (database name, user name and password, etc.) required for PHP to talk to a MySQL database. It also includes all of the table names and columns that are used in the application so if you choose to rename anything, you should only need to modify the data here.

The HSSC does use the standard PHP/MySQL connectors. If you choose to use a different database, you may need to replace this code with that which is appropriate for your platform. We have avoided using MySQL specific features, but there is no guarantee that a port to a different database will be easy.

## **Shell Functions**

There are several areas of the application that may make sense to enhance. We have provided a set of shell functions to make it easy for you to enhance the application. These functions are found in (HSCClientFuncions.php).

The three current functions are:

1. `getLegalDelay()` – which sets the delay period for how long between when a challenge is sent and when a response will be accepted by the server. Currently, the function returns 5000 for 5000 milliseconds (or 5 seconds).
2. `isScoreValid($gid,$score)` – this function allows you to check if a score for a game is acceptable... mainly, if the score is too high for the game. Currently, the function returns “true” always.
3. `cleanPlayerName($name)` – this function allows you to “clean up” proposed player names. If you do not want a naughty game to get posted, return a “” and the score will be rejected.

## Integrating the Client

The Flash portion of the High Score Security Component is provided either as an Actionscript class file or as a SWC component. The HSSC is pretty simple to use. Simply install, include, instantiate, and call. The current version is implemented in Actionscript 2.

### Usage

In the simplest scenario, simply include the HSSC Actionscript file(s) in the directory with the target application (currently, `HSLclient.as` and `MD5_2.as`). The MD5 file is separate at the moment since it is provided under a BSD license.

Second, import the client application into the game application:

```
import HSLclient;
```

Third, create an instance of the HSSC:

```
var client:HSLclient = new HSLclient(); // creates the HSSC instance
client.reset(); // resets the instance (just good coding practice, do
before you post any score.
```

Finally, post the score:

```
var hu:String = "http://www.YOURDOMAIN.com/HSL.php"; // URL of high
score processor at server

var gn:String= gameID; //ID string for the game
var gk:String = gameKey;//Key string for the game
var un:String = UserName; // Reads the Username String
var hs:String = HighScore; // Reads the score. Note, currently, the
score will only be processed at the server if it is a number.

client.postScore(hu,gn,gk,hs,un);
```

Call this function whenever you would like to post a score.

## Configuring a Game

Each game title needs to have a Game ID, a Game Name, and a Game Key. Currently, do this via your database management application. The Game ID and the Game Key need to also be stored in the Flash application. This should be done via the game’s source code. DO NOT USE

FlashVars or other external method to store the GameID and Key. The GameID and Key are both simple strings. The system has not been tested with non-ASCII formats, so use UNICODE or other string formats with great caution.

## **Sample Reports**

There are two sample report pages (HighScorePage.php and HSR.php) provided that allow you to query information about games by title, high scores for a given game, high scores for a player, or high scores for players within a game. Both rely on a set of query functions implemented in (HighScoreFunctions.php).