AhnLab HackShield Pro for Online Game

Quick Guide

V 2.0

Basic Functions

AhnLab

1 Before Getting Started

- This document is a guideline for the application of HackShield Pro SDK (or HackShield.) It has been prepared for C/C++ language in Microsoft Visual C++ 6.0 environment, and covers basic functions of HackShield.
- For easier application of HackShield codes, this document provides brief overview of HackShield Pro SDK and simple sample codes. Each game developer can use HackShield Pro SDK according to its policies.
- For reference, MiniA sample code is provided under \Sample folder.
- *"AhnLab HackShield Pro Programming Guide"* document is provided under \Doc folder. This document provides general structure of HackShield Pro and describes features and API functions. For more information, refer to this document.

2 Composition of HackShield Pro SDK

- \Bin: Files related to execution of HackShield
- \Doc: User guide
- \Include: Include file required for HackShield application
- \Lib: Library files required for HackShield application
- \PatchSet: Patch set file required for update
- \Sample: Sample files for user support

3 Preparation

- Add libraries.
  - ✓ HackShield uses two basic Windows libraries – version.lib and winmm.lib. Add these two library files.
  - ✓ \Lib\HShield.lib Library file is necessary. Copy the file and add it to the project.
- \Include\HShield.h header file is necessary. Copy the file to a proper folder.
- A license key consisting of (four-digit) game code and 24-digit character license key is needed. Each game has unique license key.

4 Application of HackShield Service

❖ Include HShield.h file previously copied in a proper folder.

```
#include “HShield.h”
```

❖ Add a function to entry-point function of game in order to call HackShield service.

✓ HackShield code is applied in order of initialization → start → game execution → stop → uninitialization. Before initializing the game client, initialize the start HackShield module. When terminating the HackShield module, stop and uninitialize the module.

```
WinMain (…)
{
    if ( !HS_Init() )
    {
        HS_UnInit();                                    HackShield Initialize
        return 0;
    }

    if ( !HS_StartService() )
    {
        HS_StopService();
        HS_UnInit();                                    HackShield Service Start
        return 0;
    }

    // Game execution routine
    …

    HS_StopService();                                   HackShield Service Stop

    HS_UnInit();                                        HackShield Uninitialize
}
```

✓ If Uninitialize is not properly called, the code may not be properly executed. An exception during game execution and calling of Uninitialize(Stop) shall be considered for the case when Initialize(Start) fails.

✓ It is recommended that the HackShield service should be started immediately after HackShield, is initialized in order to keep the game client safe.

❖ Write HackShield initialization function (_AhnHS_Initialize).

```
BOOL HS_Init()
{
    int             nRet = 0;
    TCHAR           szFullFilePath[MAX_PATH];
    TCHAR           szMsg[MAX_PATH];
    DWORD           dwOption = 0;

    // ① Define EhSvc.dll path under HackShield folder.
    lstrcat ( szFullFilePath, _T( "\\HShield\\EhSvc.dll" ) );

    // ② Define option flag to be used to call   _AhnHS_Initialize function
    dwOption = AHNHS_CHKOPT_ALL |
               AHNHS_USE_LOG_FILE |
               AHNHS_ALLOW_SVCHOST_OPENPROCESS |
               AHNHS_ALLOW_LSASS_OPENPROCESS |
               AHNHS_ALLOW_CSRSS_OPENPROCESS |
               AHNHS_DONOT_TERMINATE_PROCESS;

    // ③ Call _AhnHS_Initialize function and initialize HackShield service.
    nRet = _AhnHS_Initialize ( szFullFilePath,
                               HS_CallbackProc,              // callbackfunction
                               1000,                         // game code
                               "B228F291B7D7FAD361D7A4B7",   // License key
                               dwOption,                     // option flag
                               AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL );

    // ④ Check the return data of _AhnHS_Initialize function and consider as an error.
    if ( nRet != HS_ERR_OK )
    {
        switch( nRet )
        {
        case HS_ERR_COMPATIBILITY_MODE_RUNNING:
        case HS_ERR_NEED_ADMIN_RIGHTS:
        case HS_ERR_INVALID_FILES:
        case HS_ERR_INIT_DRV_FAILED:
        case HS_ERR_DEBUGGER_DETECT:
        default:
            wsprintf( szMsg, "An error occurred in hacking prevention.(%x)", nRet );
            break;
        }
        MessageBox( NULL, szMsg, szTitle, MB_OK );
        return FALSE;
    }
    return TRUE;
}
```

① Set the absolute path of EhSvc.dll file of HackShield Pro SDK in szFullFilePath parameter.

This file is usually located under HShield folder to be created in the executable file path. It is recommended to use GetModuleFileName function to properly set the absolute path. GetCurrentDirectory function may not get a desired path.

② The flag defined here is used as a fifth factor when calling _AhnHS_Initialize function. Set the basic functions of HackShield. In order to debug the game, delete a few options before execution. For more information about each flag, see *AhnLab HackShield Programming Guide: 2.3. Application Programming Interface"*. It is recommended to use flags used in test environment for distribution, unless there are special issues.

③ Calling _AhnHS_Initialize function requires the issued game code and the license key. The last factor sets the speed hack detection sensitivity. Speed hack detection sensitivity depends on the game developer's policies. In most cases, AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL is used. For more information about each factor, see *"AhnLab HackShield Programming Guide: 2.3. Application Programming Interface"*.

④ Check the return data of _AhHS_Initialize function. If the execution is not normal (HS_ERR_OK), the error code will display the corresponding error message. Error message for each case shall be written considering error features and the game developer's policies in compliance with *"AhnLab HackShield Programming Guide"*.

ex) In case of HS_ERR_INVALID_FILES:

```
wsprintf( szMsg, "An invalid file has been installed. Install \nProgram again. (%x)", nRet );
break;
```

❖ Write HackShield service starting function (_AhnHS_StartService).

```
BOOL HS_StartService()
{
    int             nRet = 0;
    TCHAR           szMsg[MAX_PATH];

    // ① Start the HackShield service by calling _AhnHS_StartService function.
    nRet = _AhnHS_StartService();

    // ② Check the return data of _AhnHS_StartService function and create an error.
    if ( nRet != HS_ERR_OK )
    {
        switch ( nRet )
        {
        case HS_ERR_START_ENGINE_FAILED:
        case HS_ERR_DRV_FILE_CREATE_FAILED:
        case HS_ERR_REG_DRV_FILE_FAILED:
        case HS_ERR_START_DRV_FAILED:
        default:
            wsprintf ( szMsg, "An error occurred in hacking prevention.(%x)", nRet );
            break;
        }
        MessageBox( NULL, szMsg, szTitle, MB_OK );
        return FALSE;
    }
    return TRUE;
}
```

① In order to start the HackShield service, call the Initialize function. Hacking prevention is made after StartService function is called. It is recommended to start HackShield service before the game initialization routine is executed.

② Check the return data of _AhHS_StartService function. If the result is not normal (HS_ERR_OK), a corresponding error message for the error code shall be displayed and the program shall be terminated. For error messages, see *"AhnLab HackShield Programming Guide"*.

❖ Write HackShield service stopping function. (_AhnHS_StopService)

```
BOOL HS_StopService()
{
	Int   nRet = 0;

	// ① Stop the HackShield service by calling _AhnHS_StopService function.
	nRet = _AhnHS_StopService();

	if ( nRet != HS_ERR_OK )
	{
		return FALSE;
	}
	return TRUE;
}
```

① In order to stop the HackShield service, the HackShield service must be running. Call _AhnHS_StopService function. Then, the HackShield service will stop stopping the hacking prevention function and the hacking tool detection function.

❖ Write the HackShield service uninitializing function. (_AhnHS_Uninitialize)

```
BOOL HS_UnInit()
{
    int   nRet = 0;

    // ① Stop the HackShield service by calling _AhnHS_Uninitialize function.
    nRet = _AhnHS_Uninitialize();

    if ( nRet != HS_ERR_OK )
    {
        return FALSE;
    }
    return TRUE;
}
```

① If the game client program is abnormally terminated, the HackShield hacking blocking driver may not be unloaded. Stop the service using Uninitialize. When normally terminating the client or making exceptions for normal/abnormal termination after calling of the callback function, Uninitialize shall be executed.

② For abnormal uninitialization, add the following code at the beginning of the game in order to execute the HackShield stopping routing when the game is terminated.

```
    ::SetUnhandledExceptionFilter ( Game_UnhandledExceptionHandler );
```

③ Call HackShield StopService and Uninitialize in Game_UnhandledExceptionHandler() function registered above. This is for the case when an exception occurred only.

❖ Write event transmission function related to hacking prevention and hacking tool detection functions. (_AhnHS_Callback)

```
int __stdcall HS_CallbackProc ( long lCode, long lParamSize, void* pParam )
{
	TCHAR		szMsg[MAX_PATH];

	// ① Display a proper error message for each case
	switch ( lCode )
	{
	// ② Engine Callback
	case AHNHS_ENGINE_DETECT_GAME_HACK:
		wsprintf( szMsg, “Following programs and games cannot be executed together. (%x) \n [%s]”,
			lCode, (LPTSTR)pParam );
		MessageBox( NULL, szMsg, szTitle, MB_OK );
		break;
	// ③ AutoMacro Detection
	case AHNHS_ACTAPC_DETECT_AUTOMACRO:
		wsprintf(szMsg, _T("A behavior suspected as Macro has been detected. (Code = %x"),
			lCode);
		MessageBox(NULL, szMsg, szTitle, MB_OK);
		break;
	// ④ No special processing
	case AHNHS_ACTAPC_DETECT_AUTOMOUSE:
	case AHNHS_ACTAPC_DETECT_ALREADYHOOKED:
		break;
	// ⑤ Speed
	case AHNHS_ACTAPC_DETECT_SPEEDHACK:
	case AHNHS_ACTAPC_DETECT_SPEEDHACK_APP:
		wsprintf( szMsg, “A behavior suspected as speed hack has been detected. (%x)”, lCode );
		MessageBox( NULL, szMsg, szTitle, MB_OK );
		break;
	// ⑥ Debugging Prevention
	case AHNHS_ACTAPC_DETECT_KDTRACE:
	case AHNHS_ACTAPC_DETECT_KDTRACE_CHANGED:
		wsprintf( szMsg, “Debugging attempt for the game has been detected. (%x)”, lCode );
		MessageBox( NULL, szMsg, szTitle, MB_OK );
		break;
	// ⑦ Other Abnormal Hacking Prevention Functions
	case AHNHS_ACTAPC_DETECT_DRIVERFAILED:
	case AHNHS_ACTAPC_DETECT_HOOKFUNCTION:
	case AHNHS_ACTAPC_DETECT_MESSAGEHOOK:
	case AHNHS_ACTAPC_DETECT_MODULE_CHANGE:
		wsprintf( szMsg, “An error occurred in the hacking prevention function. (%x)”, lCode );
		MessageBox( NULL, szMsg, szTitle, MB_OK );
		break;
	}
	return 1;
}
```

① Processes the data detected by HackShield. Defines each case in compliance with the game rules. For more information about each case, see *"AhnLab HackShield Programming Guide: 2.3. Application Programming Interface"*.

② This is when a hacking program or a potentially harmful program was detected. pParam notifies the name of the detected program to the user. After this message is generated, the game client shall be initialized.

③ This is when HackShield hacking tools related to AutoMacro is detected. Terminate the game client.

④ It is recommended that a separate message box or error message should not be created. If necessary, logs may be created. In case of AHNHS_ACTAPC_DETECT_ALREADYHOOKED, some APIs are hooked already and hooking may be made in some normal programs including security programs. In case of AHNHS_ACTAPC_DETECT_AUTOMOUSE, callback will not be called because it is the protection function, not the detection function.

⑤ This is when the time changing speed is abnormal like the speed hack. It is highly likely that it is the speed hack. Generate messages, and judge whether termination has been made depending on the game developer's policies.

⑥ This is when debugger trace occurred. It is highly likely that debugging is currently being made for the game program. Generate the message and stop the game client.

⑦ This is when an error occurred in message hooking or module change function. Generate the message and stop the game client.

5 Notes on Application

- Initialization options can be selectively used depending on the game developer's policies. It is recommended to use the options used in the example in order to keep the game client safe.
- It is recommended to display error messages with error codes for debugging and customer support.
- MessageBox(NULL, …) is used for messages. However, on the actual full-screen game, the message may be hidden behind the game. It is recommended to display the messages on the game. In case of using MessageBox function, use Handle instead of Null.
- Simple messages may allow the user to continue the game. Generate messages and forcibly end the game client.

6 Test and Distribution

- After applying HackShield codes, compile the project and make a game client.
- Create HShield folder under game client folder.
  - ✓ ex) …\[Game Directory]\HShield
- Copy files under \Bin\HShield folder to HShield folder.
- Check whether HackShield normally functions by executing a game client and conducting a simple test.
  - ✓ Case 1. Execute Process Explorer and check whether dll information is displayed.
  - ✓ Case 2. Rename EhSvc.dll file in HShield, and check whether it is detected.
  - ✓ Case 3. Start the hacking tool and check whether it is accurately detected in the game client.
- If no error is detected, pack the game execution file.
  - ✓ Prevent easy debugging of the game execution file using the packer.
  - ✓ HackShield provides a basic upx packer.
  - ✓ ex) upx.exe –f Source.exe –o Output.exe
- If there is no error, server interfaces are recommended. (See Quick Guide for Server Interface.)
- Conduct initial distribution as follows:
  - ✓ Application and Test ⇒ Quality Assurance (by AhnLab) ⇒ QA Report submission, review, and preparation of finial version for distribution ⇒ Distribution
  - ✓ It is recommended to delete the hshield.log file created during the test before distribution.