AhnLab HackShield

Quick Guide

V 2.0

基本機能

1. はじめに

- 本書は HackShield Pro SDK を適用するためのガイドラインです。Microsoft Visual C++ 6.0 環境において C/C++ 言語でプログラム作成することを基準とし、HackShield の基本機能について説明します。
- 初めて HackShield をプログラムコードに適用する際の概略説明および簡略化したサンプルコードが収められています。コード実装に関する詳細については、ゲーム開発会社のポリシーに沿って進行して下さい。
- ¥Sample フォルダ内に MiniA というサンプルコードが収められています。コードを適用する際に参考にして下さい。
- ¥Doc フォルダ内にある「AhnLab HackShield Pro プログラミングガイド」は HackShield Pro の全体構造や機能、API 関数の使用法についてのプログラミングガイドです。項目別の詳細についてはプログラミングガイドを参照して下さい。
- 本書に記載されているすべての製品名は該当各社の標章、商標または登録商標です。

2. HackShield Pro SDK の構成

- ¥Bin : HackShield 実行関連ファイル
- ¥Doc : 使用説明書
- ¥Include : HackShield を適用するためのインクルードファイル
- ¥Lib : HackShield を適用するためのライブラリファイル
- ¥PatchSet : アップデートするためのパッチセットファイル
- ¥Sample : ユーザーサポートのためのサンプルファイル

3. 事前準備事項

- ライブラリを追加します。
  - ✓ HackShield では Windows 基本ライブラリである version.lib、winmm.lib を使用してライブラリファイルをプロジェクトに追加します。
  - ✓ ¥Lib¥HShield.lib ライブラリファイルを適当な場所にコピーした後、プロジェクトに追加します。
- ¥Include¥HShield.h ヘッダーファイルを適当な場所にコピーします。
- ライセンス キーの発給を受けます。ライセンス キーはゲームコード（4 桁の数字）と 24 桁の文字列からなるゲームごとに固有な値です。

4. HackShield サービスの適用

- ゲームプロジェクトに先ほどコピーした HShield.h ファイルをインクルードします。

```
#include "HShield.h"
```

- ゲームのエントリーポイント関数に HackShield サービスを呼び出す関数を追加します。
  - ✓ HackShield コードは、初期化（Initialize）→ 開始（Start）→ ゲーム実行 → 停止（Stop）→ 完了（Uninitialize）の順に適用します。ゲームクライアントを初期化する前に HackShield モジュールの初期化と開始を行い、終了時には必ず停止と完了を行って下さい。

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

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

    //ゲーム実行ルーチン
    …

    HS_StopService();

    HS_UnInit();
}
```

HackShield Initialize

HackShield Service Start

HackShield Service Stop

HackShield Uninitialize

- ✓ Uninitialize が正しく呼び出されないと、ゲームを再実行する際に正常に起動できないことがあります。Initialize（Start）に失敗した時に処理する部分やゲーム実行中に例外処理をする部分で必ずUninitialize（Stop）を呼び出して下さい。
- ✓ ゲームクライアントを安全に保護するため、HackShield の初期化が終わったらすぐに HackShield サービスを開始することを推奨します。

❖ HackShield 初期化関数（_AhnHS_Initialize）を作成します。

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

      //①HackShield フォルダの EhSvc.dll の位置を指定します。
      lstrcat ( szFullFilePath,_T( "¥¥HShield¥¥EhSvc.dll" ) );

      //②_AhnHS_Initialize 関数の呼び出しに使用するオプションフラグを定義します。
      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_AhnHS_Initialize 関数を呼び出して HackShield サービスを初期化します。
    nRet = _AhnHS_Initialize ( szFullFilePath,
                            HS_CallbackProc,        //コールバック関数
                            1000,    //ゲームコード
                            "B228F291B7D7FAD361D7A4B7",     //ライセンス キー
                            dwOption,       //オプションフラグ
                            AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL );

    //④_AhnHS_Initialize 関数の戻り値を調べてエラー処理します。
    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,"ハッキング防止機能に問題が発生しました。(%x)",nRet );
      break;
    }
    MessageBox( NULL,szMsg,szTitle,MB_OK );
    return FALSE;
  }
  return TRUE;
}
```

① szFullFilePath 変数には HackShield SDK の EhSvc.dll ファイルの絶対パスを設定します。このファイルはデフォルトでは実行ファイルパスに生成される Hshield フォルダ内に格納されています。正しい絶対パスを設定するには GetCurrentDirectory 関数ではなく、GetModuleFileName 関数を使用することをお勧めします。

② HackShield の基本的な機能を設定します。ここで定義したフラグは _AhnHS_Initialize 関数を呼び出す時に 5 番目の引数に使われます。ゲームデバッグのためにはいくつかのフラグを削除し、実行する必要があります。各フラグに関する詳細は「AhnLab HackShield プログラミングガイド : 2.3. Application Programming Interface」を参照して下さい。特別な問題がなければ、テスト環境および配布時には上記サンプルのフラグを使用することを推奨します。

③ _AhnHS_Initialize 関数を呼び出すにはゲームコードとライセンス キーが必要です。最後の引数にはスピードハッキング検知に対するレベルを設定します。スピードハッキングの検知レベルはゲーム会社のポリシーに従って決定して下さい。一般的には AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL を使用します。各引数に関する詳細は「AhnLab HackShield プログラミングガイド : 2.3. Application Programming Interface」を参照して下さい。

④ _AhHS_Initialize 関数の戻り値が正常実行（HS_ERR_OK）ではなかった場合、エラーコードに従ってエラーメッセージを出力して終了します。エラーメッセージは「AhnLab HackShield プログラミングガイド」を参照の上、エラーの特性とゲーム開発会社のポリシーに従って作成して下さい。

ex) HS_ERR_INVALID_FILES の場合 :

wsprintf(szMsg,"正しくないファイルがインストールされました。¥n プログラムを再インス

トールして下さい。(%x)",nRet);
break;

❖ HackShield サービス開始関数（_AhnHS_StartService）を作成します。

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

      //①_AhnHS_StartService 関数を呼び出して HackShield サービスを開始します。
      nRet = _AhnHS_StartService();

      //②_AhnHS_StartService 関数の戻り値を調べてエラー処理します。
      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,"ハッキング防止機能に問題が発生しました。(%x)",nRet );
              break;
         }
         MessageBox( NULL,szMsg,szTitle,MB_OK );
         return FALSE;
      }
      return TRUE;
}
```

① HackShield サービスを開始するには Initialize 関数を呼び出す必要があります。ハッキングに対する防御は StartService 関数が呼び出された後に成立するため、ゲーム初期化ルーチンを実行する前に HackShield サービスを開始することを推奨します。

② _AhHS_StartService 関数の戻り値が正常実行（HS_ERR_OK）ではなかった場合、エラーコードに従って適切なエラーメッセージを出力し、プログラムを終了します。エラーメッセージは「AhnLab HackShield プログラミングガイド」を参照の上、エラーの特性に合わせて作成します。

❖ HackShield サービス停止関数を作成します。(_AhnHS_StopService)

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

      //①_AhnHS_StopService 関数を呼び出して HackShield サービスを停止します。
      nRet = _AhnHS_StopService();

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

① HackShield サービスの停止は HackShield サービスが動作中にのみ可能です。_AhnHS_StopService 関数を呼び出すと HackShield サービスが停止し、ハッキング遮断機能とハッキングツール検出機能を停止します。

❖ HackShield サービス終了関数を作成します。(_AhnHS_Uninitialize)

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

      //①_AhnHS_Uninitialize 関数を呼び出して HackShield サービスを終了します。
      nRet = _AhnHS_Uninitialize();

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

① ゲームクライアントのプログラムが正常終了しなかった場合、HackShield ハッキング遮断ドライバをアンロードできないことがあります。このような場合にプログラムを再実行すると正常に起動されないことがあるため、サービスを終了する時には必ず Uninitialize を実行してください。クライアントの正常終了以外にも、コールバック関数呼び出し後の終了や正常終了しなかった場合の例外処理でも Stop と Uninitialize を必ず実行して下さい。

❖ ハッキング遮断機能とハッキングツール検出機能に関連したイベント通知関数を作成します。
(_AhnHS_Callback)

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

      //①状況に合わせてエラーメッセージを出力します。
      switch ( lCode )
      {
      //②Engine Callback
      case AHNHS_ENGINE_DETECT_GAME_HACK:
         wsprintf( szMsg,"次のプログラムとゲームを同時に実行できません。(%x) ¥n [%s]",
                 lCode,(LPTSTR)pParam );
         MessageBox( NULL,szMsg,szTitle,MB_OK );
         break;

      //③オートマクロ検出
      case AHNHS_ACTAPC_DETECT_AUTOMACRO:
         wsprintf(szMsg,_T("マクロ機能とみられる動作が検知されました。(Code = %x"),
                 lCode);
         MessageBox(NULL,szMsg,szTitle,MB_OK);
         break;

      //④別途の処理不要
      case AHNHS_ACTAPC_DETECT_AUTOMOUSE:
      case AHNHS_ACTAPC_DETECT_ALREADYHOOKED:
         break;

      //⑤Speed 関連
      case AHNHS_ACTAPC_DETECT_SPEEDHACK:
      case AHNHS_ACTAPC_DETECT_SPEEDHACK_APP:
         wsprintf( szMsg,"スピードハッキングとみられる動作が検知されました。(%x)",lCode );
         MessageBox( NULL,szMsg,szTitle,MB_OK );
         break;

      //⑥デバッグ防止
      case AHNHS_ACTAPC_DETECT_KDTRACE:
      case AHNHS_ACTAPC_DETECT_KDTRACE_CHANGED:
         wsprintf( szMsg,"ゲームに対するデバッグの試行が検知されました。(%x)",lCode );
         MessageBox( NULL,szMsg,szTitle,MB_OK );
         break;

      //⑦その他ハッキング防止機能の異常
      case AHNHS_ACTAPC_DETECT_DRIVERFAILED:
      case AHNHS_ACTAPC_DETECT_HOOKFUNCTION:
      case AHNHS_ACTAPC_DETECT_MESSAGEHOOK:
        wsprintf( szMsg,"ハッキング防御機能に異常が発生しました。(%x)",lCode );
```

```
            MessageBox( NULL,szMsg,szTitle,MB_OK );
            break;
        }
        return 1;
}
```

① HackShield で検知された情報に関する処理を行います。ゲームルールに従ってそれぞれの状況について定義して下さい。状況に関する詳細は「AhnLab HackShield プログラミングガイド : 2.3. Application Programming Interface」を参照して下さい。

② ハッキングプログラムやゲーム進行に問題を起こす危険のあるプログラムを検知した場合です。pParam が検知したプログラム名をユーザーに知らせます。メッセージを出力し、ゲームクライアントを終了して下さい。

③ HackShield でオートマクロ関連ハッキングツールを検出した場合です。ゲームクライアントを終了して下さい。

④ この場合については別途メッセージボックスやエラーの処理をする必要はありません。必要に応じてログのみ作成します。AHNHS_ACTAPC_DETECT_ALREADYHOOKED は一部 API がすでにフッキングされている場合です。セキュリティプログラムのように正常なプログラムでもフッキングすることがあります。AHNHS_ACTAPC_DETECT_AUTOMOUSE の場合、検出ではなく保護機能であるため Callback が呼び出されません。

⑤ スピードハッキングのようにシステムの時間変化速度が正常ではない場合です。スピードハッキングである可能性が高いため、メッセージを出力し、ゲーム開発会社のポリシーに従って終了するかどうか判断します。

⑥ デバッグトレースが発生した場合です。ゲームプログラムに対するデバッグ作業が行われている可能性が高いため、メッセージを出力し、ゲームクライアントを終了して下さい。

⑦ メッセージフッキングやモジュール変更などの機能に異常が生じた場合です。メッセージを出力し、ゲームクライアントを終了して下さい。

5. 適用時の注意事項

- Initialize する時のサンプルに使われているフラグは、ゲーム開発会社のポリシーに従って適用するかどうかを選択することができますが、ゲームクライアントを安全に保護するためにはサンプルに使われているフラグをそのまま使用することを推奨します。
- デバッグおよびカスタマーサポートのためにエラーメッセージを出力する場合はエラーコードを併せて出力することをお勧めします。
- メッセージ処理時のサンプルに MessageBox（NULL,…）を使用していますが、Full-Screen の状態ではメッセージボックスがゲーム画面の背面に表示されることがあるため、ゲーム上のメッセージで処理することをお勧めします。MessageBox 関数を使用する場合、NULL ではなく Handle 処理をして下さい。
- メッセージ処理時にはメッセージを出力するだけではなく、メッセージを出力し、ゲームクライアントを強制終了して下さい。

6. テストおよび配布

- HackShield コードの適用が完了したら、プロジェクトをコンパイルしてゲームクライアントを生成します。
- ゲームクライアントのフォルダに Hshield というフォルダを生成します。
  - ex) …¥[Game Directory]¥HShield
- ¥Bin¥Hshield フォルダ内のファイルを Hshield フォルダにコピーします。
- ゲームクライアントを実行して簡単なテストを行い、HackShield の動作可否を確認します。
  - Case 1. Process Explorer を実行して dll 情報が表示されるかを確認
  - Case 2. Hshield 内にある EhSvc.dll ファイルをリネームし、これを検知するかを確認
  - Case 3. ハッキングツールを起動し、ゲームクライアントでこれを正確に検知するかを確認

- ❖ 異常がないことを確認したらゲーム実行ファイルをパッキングします。
  - ✓ パッカー（Packer）を使用することでゲーム実行ファイルが簡単にデバッグされないよう防止することができます。
  - ✓ HackShield では基本的な upx パッカーを提供しています。
  - ✓ ex) upx.exe -f Source.exe -o Output.exe
- ❖ 異常がなければサーバー連動を適用することを推奨します。(Quick Guide サーバー連動参照)
- ❖ 最初に配布する時には、次の過程に従って配布して下さい。
  - ✓ 適用およびテスト ⇒ Quality Assurance 進行（Ahnlab）⇒ QA Report を受け取り、修正事項を確認したうえで最終配布バージョンを作成 ⇒ 配布
  - ✓ テストの途中で生成された hshield.log ファイルは削除してから配布することをお勧めします。