

AhnLab HackShield

プログラミングガイド

Version 5.9

AhnLab HackShield for Online Game

はじめに

Copyright (C) AhnLab, Inc. 2002-2008. All rights reserved.

本プログラミングガイドの内容およびプログラムは、著作権法とコンピュータプログラム保護法によって保護されています。

文書の概要

本書では SDK 形態で提供する AhnLab HackShield for Online Game、AhnLab HackShield Pro for Online Game（以下 HackShield）の全体構造や機能、API 関数の使用法について説明します。

カスタマーサポート

AhnLab Customer Center	
住所	〒101-0021 東京都千代田区外神田 4-14-1 秋葉原UDX 8階北
ホームページ	http://www.ahnlab.co.jp
Email	hs_alj@ahnlab.co.jp
電話	031-772-8250
Fax	031-772-8901

目次

1. HackShieldの紹介	9
1.1. 機能	9
1.2. 動作環境	10
2. 基本機能	11
2.1. 概要	11
機能	11
特徴	12
システム構造(System Architecture)	13
2.2. Application Programming	15
プログラミング手順	15
2.3. プログラミング準備	17
2.3.1 HackShield関連ファイル	17
2.3.2 プログラミング適用方法	18
HackShield update関数呼び出し	18
モニタリング関数呼び出し	19
ライセンスキーの発行	19
初期化関数呼び出し	20
サービス開始関数呼び出し	22
Callback関数の作成	23
サービス停止関数呼び出し	24
サービス終了関数呼び出し	25
HackShieldログ収集関数呼び出し	26
2.4. Application Programming Interface	28
_AhnHS_Initialize	28
_AhnHS_Callback	38
_AhnHS_StartService	43
_AhnHS_StopService	46
_AhnHS_Uninitialize	47
_AhnHS_PauseService	48
_AhnHS_ResumeService	50
_AhnHS_CheckHackShieldRunningStatus	52
_AhnHS_SendHsLog	54
3. HackShield アップデート機能	55
3.1. 概要	55
機能	55
特徴	55
システムアーキテクチャ	56
3.2. Application Programming	58
プログラミング適用方法	58
3.3. Application Programming Interface	66
_AhnHS_HSUpdateEx	66
_AhnHS_HSUpdate	70
4. 拡張サーバー連動機能	74

4.1	概要	74
	機能	74
	特徴	75
	拡張サーバー連動関連ファイル	75
	システムアーキテクチャ	76
4.2	Application Programming	80
	プログラミング適用方法	80
4.3	Application Programming Interface	83
	_AhnHS_CreateServerObject	83
	_AhnHS_CloseServerHandle	85
	_AhnHS_CreateClientObject	86
	_AhnHS_CloseClientHandle	88
	_AhnHS_MakeRequest	89
	_AhnHS_VerifyResponseEx	91
	_AhnHS_VerifyResponse	94
	_AhnHS_MakeResponse	100
5	サーバー連動クラック防止機能	104
5.1	概要	104
	機能	104
	特徴	105
	システムアーキテクチャ	105
5.2	Application Programming	108
	プログラミング適用方法	108
5.3	Application Programming Interface	111
	_AntiCpSvr_Initialize	111
	_AntiCpSvr_Finalize	113
	_AntiCpSvr_MakeGuidReqMsg	114
	_AntiCpSvr_AnalyzeGuidAckMsg	116
	_AntiCpSvr_MakeReqMsg	121
	_AntiCpSvr_AnalyzeAckMsg	124
	_AhnHS_MakeGuidAckMsg	128
	_AhnHS_MakeAckMsg	130
	_AhnHS_SaveFuncAddress	132
6	モニタリングサービス機能	134
6.1	概要	134
	機能	134
	特徴	134
	システムアーキテクチャ	135
6.2	Application Programming	136
	Programming Application	136
6.3	Application Programming Interface	139
	_AhnHS_StartMonitor	139
	_AhnHS_SetUserId	141
7	LMP機能	142
7.1	概要	142
	機能	142
	特徴	143
	システムアーキテクチャ	143

7.2	Application Programming	144
	プログラミング適用方法	145
	_AhnHS_IsModuleSecure	147
	_AhnHS_IsModuleSecure	148
8	その他の機能	150
8.1	データファイル/メッセージ暗号化機能	150
8.1.1	概要	150
	機能	150
	特徴	150
	システムアーキテクチャ	151
8.1.2	Application Programming	152
	プログラミング手順	152
	プログラミング準備	152
	HsCryptLib ファイル	152
	_HsCrypt_InitCrypt	153
	_HsCrypt_GetEncMsg	154
	_HsCrypt_GetDecMsg	154
	_HsCrypt_FRead	155
8.1.3	Application Programming Interface	156
	_HsCrypt_InitCrypt	156
	_HsCrypt_GetEncMsg	158
	_HsCrypt_GetDecMsg	159
	_HsCrypt_FRead	160
8.2	ユーザー権限実行サポート機能	163
8.2.1	概要	163
	機能	163
	特徴	163
	システムアーキテクチャ	164
8.2.2	Application Programming	165
	プログラミング手順	165
	プログラミング準備	166
	_AhnHsUserUtil_CreateUser	166
	_AhnHsUserUtil_SetFolderPermission	167
	_AhnHsUserUtil_DeleteUser	168
	_AhnHsUserUtil_IsEnableHSAAdminRights	168
	_AhnHsUserUtil_CheckHSShadowAccount	168
	_AhnHsUserUtil_IsAdmin	169
8.2.3	Application Programming Interface	170
	_AhnHsUserUtil_CreateUser	170
	_AhnHsUserUtil_SetFolderPermission	172
	_AhnHsUserUtil_DeleteUser	174
	_AhnHsUserUtil_IsEnableHSAAdminRights	175
	_AhnHsUserUtil_CheckHSShadowAccount	176
	_AhnHsUserUtil_IsAdmin	177
9	ツールの使い方	178
9.1	AntiCpSvr ツール	178
	機能	178
9.2	AntiCpSvrツールの使い方	179
	UI 手動設定基盤のCRC情報ファイル生成	179

CRC 情報ファイルの自動生成	180
9.3 HSBGenツール(HackShield 専用、4.2 以降のバージョン)	181
機能	181
9.4 HSBGen ツールの使い方	182
UI 手動設定基盤のHSB情報ファイル生成	182
HSB 情報ファイルの自動生成	184
HSBGen.ini 説明	187
9.5 HSUpSetEnvツール(HackShield専用, 5.1以後バージョン)	191
機能	191
9.6 HSUpSetEnvツールの使い方	192
HSUpdate.env ファイル作成	192
9.7 CSInspector ツール	194
機能	194
対象のファイルを保護対象に設定	194
9.8 SetServerListツール(HackShield専用, 5.1以後バージョン)	195
機能	195
9.9 SetServerListツールの使い方	196
afs.dat ファイル作成	196
afs.dat ファイル配布	197
9.10 HSBHelperツールの使い方	198
機能	198
10 付録	200
10.1 FAQ	200

List of Tables

表 2-1 HackShield ファイル.....	19
表 2-2 ゲームアップデートサーバーに必要なファイル	20
表 2-3 [AHNHS_CHKOPT_SELF_DESTRUCTION利用時のコールバックコード].....	34
表 3-1 アップデート関連ファイル	64
表 3-2 アップデートサーバーの必須ファイルリスト.....	65
表 3-3 アップデートに使われるデータファイル	6660
表 3-4 HackShieldアップデートサーバースペック.....	66
表 4-1 サーバー連動機能バージョン管理.....	85
表 4-2 AntiCpXSvr関連ファイル	88
表 5-1サーバー連動バージョン管理	117106
表 5-2 AntiCPSvr関連ファイル	10119
表 6-1 Monitoring関連ファイル.....	148136
表 7-1 LMP関連ファイル	145157
表 7-2 LMPサポートパッカー	149162
表 8-1 HsCryptLibファイル	152165
表 8-2 HsUserUtilファイル	166182

List of Figures

図. 2-1 HackShield システムアーキテクチャ	14
図. 2-2 アプリケーションプログラムの手順	17
図. 3-1 HackShield アップデート	63
図. 3-2 デフォルトHackShield アップデートイメージ	71
図. 4-1 AntiCpXの動作	86
図. 5-1 AntiCpSvr の動作	107
図. 6-1 モニタリングサービスの動作	147
図. 7-1 Local Memory Protection	144
図. 8-1 HsCryptLibの全体構造と動作原理	151
図. 8-2 HsUserUtilの全体構造と動作原理	163
図. 8-3 HsUserUtilプログラミング手順	165
図. 9-1 HackShield CRC 情報生成ツール	179
図. 9-2 Command-line タイプの AntiCpSvrTool.exe	1198
図. 9-3 HSB File Generator	1201
図. 9-4 Command-line Type HSBGen.exe (一般ファイルの場合)	1203
図. 9-5 Command-line Type HSBGen.exe (パッキングされたファイルの場合)	1203
図. 9-6 Command-line Type HSBGen.exe(パッキングされたファイルを実行する場合) ...	1203
図. 9-7 Command-line Type HSBGen.exe (パッキングされたファイル実行時、引数が必要 な場合)	1204
図. 9-8 HSUpSetEnv.exe	1922
図. 9-9 CSInspector.exe	19213
図. 9-10 SetServerList.exe	196215
図. 9-11 SetServerList ツールのIPアドレス入力ウィンドウ	197216
図. 9-12 HSBHelper Tool	19217

1. HackShield の紹介

HackShield は Ahnlab,Inc. が開発したソリューションであり、ゲームプログラムのハッキングツール検出、ハッキング遮断、クラック防止、実行ファイルのリアルタイムプロテクト、データの暗号化などの機能を提供します。

1.1. 機能

ハッキング遮断機能

シグネチャベース、メモリーヒューリスティックの2つのスキャン方式により、ハッキングツール検出、メモリハッキング遮断、スピードハック遮断、デバッグング遮断、メッセージフックング遮断、ファイルの偽造/改ざんの遮断、オートマウス遮断などを行います。

サーバー連動クラック防止機能

サーバーと連動してリアルタイムでの実行ファイル操作検知、メモリ操作検知、HackShield の動作状態確認などを行います。

データファイル/メッセージの暗号化機能

主要データファイルやサーバー、クライアント間で通信するメッセージを暗号化し、データファイルやメッセージが流出しても内容を確認できないように保護します。

User 権限実行機能

NT 系 OS で Administrator 権限ではない一般 User 権限や Guest 権限で実行できます。

1.2. 動作環境

HackShield のインストールと動作に必要なシステム環境は以下の通りです。

クライアント

項目	推奨スペック	最小スペック
OS	Windows 98/ME/ Windows 2000 Professional / Windows XP / Windows VISTA / Windows 7 / Windows Server 2003	Windows 98
CPU	Intel Pentium 500Mhz 以上	Intel Pentium 133MHz 以上 または、IBM-PC 互換機
RAM	128MB 以上	32MB
HDD	2MB以上	2MB

サーバー

OS	Platform
Windows 2000, Windows 2003 (x86, x64, IA64) Windows Server 2008	x86, x64
Solaris 8, 10 (32bit)	x86
Fedora Linux 7.1(x86), 11(x64), redhat enterprise 4, cent OS 5.3	x86, x64

注意

Web サーバー、DB サーバーなどサーバー用途の Windows NT 系 OS で HackShield を実行する場合、予想外の性能低下などの問題が発生することがあります。

参考

CPU は Intel x86 系 (x86 互換 AMD 系を含む) Windows NT を実行する DEC Alpha
や NEC pc 98xx machine OS についてはサポートの対象となりません。

2. 基本機能

HackShield のハッキング遮断およびハッキングツール検出機能を実行するのに必要なシステム構造とプログラミング方法について説明します。

参考

本書で使用しているサンプルコードは Microsoft Visual C++ 6.0 を対象とした C/C++ 言語により作成されています。プログラミングに使われている言語は各プログラムの特性やシステム環境によって変更することができます。

2.1. 概要

機能

エンジンベース(シグネチャ & メモリーヒューリスティック)のハッキングツール検出機能

Ahnlab,Inc. のアンチウイルス技術を使いエンジンにシグネチャが登録されているハッキングツールを検出します。ハッキングツールのシグネチャがエンジンに登録されていると、該当するプロセスを強制終了して（オプションによっては強制終了しないことがあります）Callback 関数でファイル名をゲームクライアントに通知します。エンジンに登録されていないハッキングツールが発見されると新しいハッキングツールに関するシグネチャが追加され、エンジンをアップデートすることで新しいハッキングツールを遮断することができます。

メモリハッキング遮断機能

ゲームプログラムで使用するメモリを Windows API（OpenProcess、Read/WriteProcessMemory・・・）からのアクセスを遮断します。カーネルレベルで直接メモリをトレースし、結果値を操作するなどのハッキングを遮断します。

注意

ハッキングツールではなくてもメモリに直接アクセスするプログラムの場合、正常に動作しないことがあります。

スピードハック遮断機能

スピードハックには大きく分けて、システムに装着されているタイマーを操作するハードウェア方式と OS の日付関連 API を操作するソフトウェア方式があります。スピードハックを遮断するためにはマイクロプロセッサレベルにおいて生じるシステムの時間と OS の時間の論理的な差を常を確認します。一定以上の時間差が生じると Callback 関数でゲームクライアントに通知します。

ユーザーのシステムや OS、ゲームの性質によってスピードハックの許容度に差があるため、パラメータにより別途、スピードハック検知率に関するレベルを 5 段階で設定することができます。

参考

スピードハックとは、タイマー処理マイクロプロセッサや Windows システムで提供している時間関連関数を操作して時間関連機能が正常に動作できないようにするプログラムやハッキングの手法をいいます。

デバッグ遮断

デバッガを使ってゲームプログラムを分析し、ハッキングを試行できないようにすべてのデバッガのデバッグトレーシングを遮断します。デバッグトレーシングの試行が検知されると Callback 関数によってゲームクライアントに通知します。HackShield を初期化する際にも、まずは SoftICE のようなデバッグプログラムが実行中であることを確認し、デバッグプログラムが実行されていたらエラーを返します。

オートマウス遮断

マウス入力を自動的に処理してゲームプログラムを操作し、ゲームサーバーに過負荷を与えることのないようオートマウスを遮断します。

特徴

インターフェイス関数(API)の提供

HackShield の機能を使った実行結果値を確認できるようにインターフェイス DLL を提供します。ゲーム開発会社ではインターフェイス DLL を使い、固有ポリシーに従って HackShield のハッキング遮断機能から必要な機能のみを実行するようにプログラミングすることができます。

サンプルプログラムの提供

HackShield の API を使って実行したテスト用ゲームクライアントのプログラムを提供します。HackShield の機能や性能を確認し、クライアントプログラムを開発するのにサンプルプログラムが役立ちます。

ハックシールドアップデートの提供

ハックシールドアップデートは最新のハッキング遮断機能とハッキングツール検出エンジンを提供します。ゲームクライアントプログラムは、アップデートサーバーからファイルンをダウンロードすることでハッキング対応能力を常に最新の状態に保つことができます。

システム構造(System Architecture)

HackShield は独立した実行ファイル形態ではない SDK を使ったライブラリ形態で提供します。HackShield の全体構造、動作概要原理は次のとおりです。:

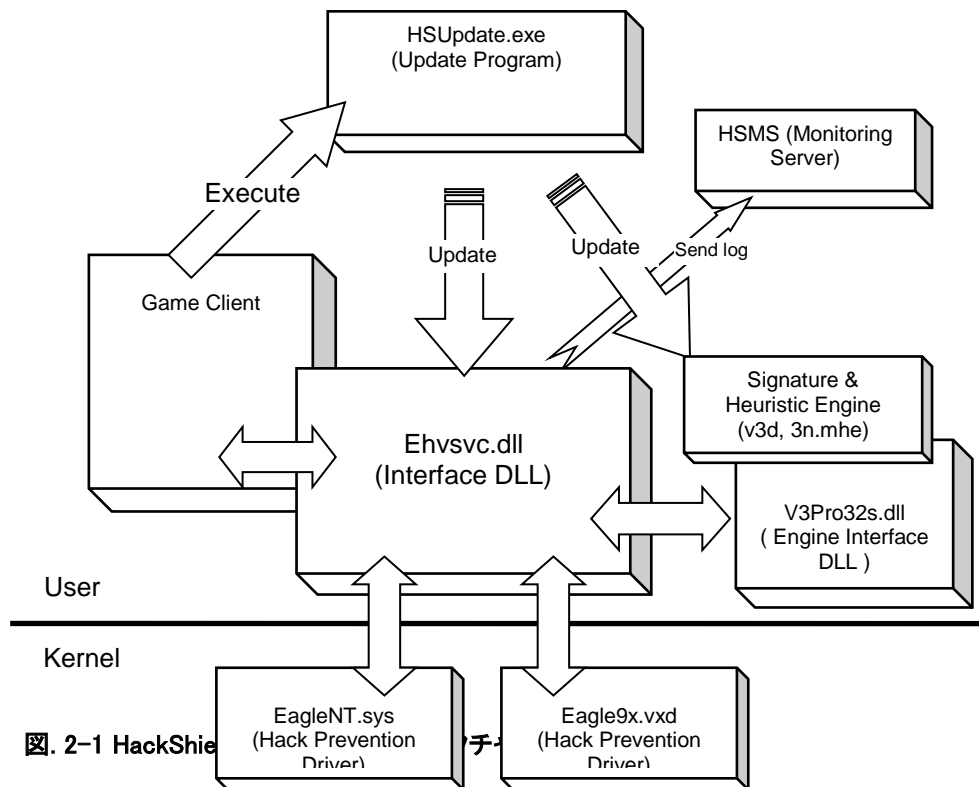


図. 2-1 HackShield

HSUpdate.exe (アップデートプログラム)

HackShield 専用アップデートプログラムとしてアップデートサーバーからハッキング遮断モジュールやハッキングツール検出モジュールを含むエンジンファイルをダウンロードしてアップデートします。エンジンファイルアップデートのみ使用するように選択もできます。

EhSvc.dll (インターフェイス DLL)

インターフェイスファイルとしてハッキング遮断モジュールやハッキングツール検出エンジンを動作させます。ハッキングツール検出やハッキング遮断結果を Callback 関数によってゲームクライアントに通知します。

V3Pro32s.dll (エンジンインターフェイス DLL)

Ahnlab,Inc. のアンチウイルス技術を使ったハッキングツール検出エンジンを動作させます。エンジンにシグネチャが登録されているハッキングツールが実行されると、直ちにこれを検出してゲームプロセスに通知します。エンジンに登録されていないハッキングツールを発見すると、新しいハッキングツールに関するシグネチャをエンジンに登録して直ちに遮断し、最新のエンジンをアップデートします。

Eagle9x.vxd or EagleNT.sys(ハッキング遮断 Driver)

カーネルモードで動作するドライバファイルとしてハッキング遮断機能を実行します。EhSvc.dll (インターフェイス DLL) ファイルに含まれており、OS によって Eagle9x.vxd または EagleNT.sys がシステムにロードされます。

2.2. Application Programming

HackShield の API を使ってハッキング遮断機能およびハッキングツール検出機能を実行する方法について説明します。

参考

本書で使用しているサンプルコードは Microsoft Visual C++ 6.0 を対象とした C/C++ 言語により作成されています。プログラミングに使われている言語は各プログラムの特性やシステム環境によって変更することができます。

プログラミング手順

ハッキングツール検出およびハッキング遮断機能のプログラミング手順は以下のとおりです。:

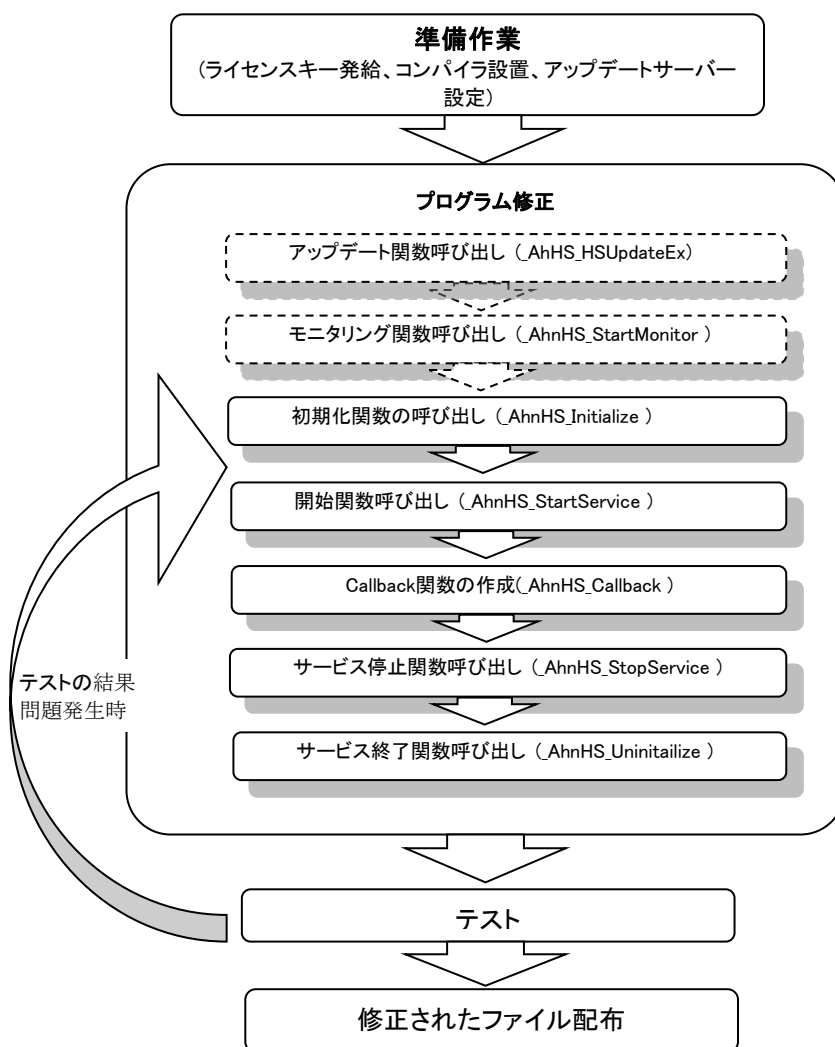


図. 2-2 アプリケーションプログラムの手順

1. **準備工程**: 提供を受けた HackShield のファイルリストを確認し、必要なファイルをコピーします。プログラミングするにはライセンスキーの発給を受ける必要があります。

2. **アップデート関数呼び出し(AhHS_HSUpdateEx)**: HackShieldアップデートを利用する場合、HackShieldモジュールを自動的に更新するためにアップデート関数を呼び出す必要があります。

参考

HackShield アップデートを使ってモジュールの更新をする場合、別途提供されているアップデートライブラリ (HSUpChk.lib) を使用してアップデート関数を呼び出します。

3. **モニタリング関数呼び出し (AhnHS_StartMonitor)**: ハックシールドモニタリングサーバー (HSMS) を設置している場合は、モニタリング関数を呼び出してハッキング情報とエラー情報をサーバーへ送信を開始します。

4. **初期化関数呼び出し (AhnHS_Initialize):** HackShieldの初期化関数を呼び出してファイルの偽造・改ざんのスキャンやデータの初期化を行います。
5. **サービス開始関数呼び出し (AhnHS_StartService):** サービス開始関数を呼び出してハッキングを遮断し、ハッキングツールを検出します。
6. **Callback関数の作成(AhnHS_Callback):** ハッキング遮断機能やハッキングツール検出機能の実行結果を処理する Callback 関数を作成します。
7. **サービス停止関数呼び出し (AhnHS_StopService):** プログラムの終了処理の部分にサービス停止関数の呼び出しを追加することでハッキング遮断機能やハッキングツール検出機能を停止させます。
8. **サービス終了関数呼び出し (AhnHS_Uninitialize):** サービス停止関数を呼び出した後にサービス終了関数を呼び出します。

注意

ゲームプログラムが正常に終了しない場合でも、HackShield の終了処理は必ず呼び出すようにプログラムを作成してください。

異常終了時の処理のため、ゲームの最初のところに以下のハックシールドの終了処理を加えてください。

```
void Game_UnhandledExceptionHandler ()
{
    _AhnHS_StopService();
    _AhnHS_Uninitialize();
}

::SetUnhandledExceptionFilter ( Game_UnhandledExceptionHandler );
```

9. 修正したソースコードが通常操作で問題が発生しないことをテストします。
10. ゲームクライアントをユーザーに配布します。

2.3. プログラミング準備

この章にはHackShield を使ってプログラミングを始める前にインストールディレクトリに含まれる HackShield のファイルリストとコンパイラの設定方法、アップデートモジュールの使用方法、アップデートサーバー設定方法、ライセンスキー発給方法について説明します。

2.3.1.HackShield 関連ファイル

インストールディレクトリ

HackShield ファイルをゲームクライアントがインストールされているフォルダの下位に HackShield フォルダを生成してインストールします。

<Game Directory>¥HShield¥

HackShield 関連ファイル

表 2-1 HackShield 関連ファイル

ファイル名	インストールフォルダ	説明
3n.mhe	[HackShield フォルダ]	ヒューリスティックエンジンファイル
EhSvc.dll	[HackShield フォルダ]	インターフェース DLL
hshield.dat	[HackShield フォルダ]	HackShield 関連 dat ファイル
v3pro32s.dll	[HackShield フォルダ]	ハッキングツール検出エンジンインターフェース DLL
asc_com.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
asc_dh.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
asc_fse.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
asc_intg.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
asc_mmgr.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
asc_unp.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
fse_base.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
fse_fact.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
fse_pe.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
gfs_base.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
gfs_fact.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
gfs_file.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
gfs_mem.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
gfs_os.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
gfs_proc.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
gfs_util.dll	[asc フォルダ]	ハッキングツール検出エンジンインターフェース DLL
0asc.scd	[asc フォルダ]	ハッキング遮断エンジンパターンファイル
0scure.scd	[asc フォルダ]	ハッキング遮断エンジンパターンファイル
0sgame.scd	[asc フォルダ]	ハッキング遮断エンジンパターンファイル
0spe3f.scd	[asc フォルダ]	ハッキング遮断エンジンパターンファイル
moduler.scd	[asc フォルダ]	ハッキング遮断エンジンパターンファイル
option.scd	[asc フォルダ]	ハッキング遮断エンジンパターンファイル
AhnRpt.exe	[HackShield フォルダ]	HackShieldログ収集ファイル
HsLogMgr.exe	[HackShield フォルダ]	HackShieldログ選択ファイル
AhnRpt.ini	[HackShield フォルダ]	HackShieldログ収集設定ファイル
HShield.h	[Game source フォルダ]	ヘッダーファイル
HShield.lib	[Game source フォルダ]	ライブラリファイル

コンパイラ設定

HackShield を使用するゲームクライアントプログラムのプロジェクトファイルに、HShield.lib ファイルをライブラリやソースコードリストに登録する必要があります。

2.3.2プログラミング適用方法

HackShieldアップデート関数呼び出し

[ケース1. HackShieldアップデートモジュール使用]

詳細は、[3. HackShieldアップデート](#)を参照してサーバーとクライアントに適用方法を確認してください。

クライアントプログラムにアップデートライブラリ関数を適用するためには、HackShieldインターフェイスDLLを呼び出す前に EhSvc.dll を読み込む必要があります。

[ケース2. ゲーム開発会社のパッチモジュール使用]

HackShieldファイルのアップデートをする際にゲーム会社独自のアップデートモジュールを利用する方法です。この方法ではゲームと別に新たな、モジュールを配布する必要がありません。ゲーム開発者は、すでに存在する自社のパッチモジュールを配布するだけです。

独自のパッチモジュールを利用する場合は、以下のHackShieldファイルを配布する必要があります。:

表 2-2 ゲームアップデートサーバーに必要なファイル

File name	Description
3n.mhe	ヒューリスティックエンジンファイル
EhSvc.dll	HackShield インターフェース DLL
hshield.dat	HackShield 関連 dat ファイル
V3pro32s.dll	ハッキングツール検出エンジンインターフェイスDLL
0asc.scd	ハッキングツールパターンエンジンファイル
0scure.scd	ハッキングツールパターンエンジンファイル
0sgame.scd	ハッキングツールパターンエンジンファイル
0spe3f.scd	ハッキングツールパターンエンジンファイル
asc_com.dll	ハッキングツール検出エンジンインターフェイスDLL
asc_dh.dll	ハッキングツール検出エンジンインターフェイスDLL
asc_fse.dll	ハッキングツール検出エンジンインターフェイスDLL
asc_intg.dll	ハッキングツール検出エンジンインターフェイスDLL
asc_mmgr.dll	ハッキングツール検出エンジンインターフェイスDLL
asc_unp.dll	ハッキングツール検出エンジンインターフェイスDLL
fse_base.dll	ハッキングツール検出エンジンインターフェイスDLL
fse_fact.dll	ハッキングツール検出エンジンインターフェイスDLL
fse_pe.dll	ハッキングツール検出エンジンインターフェイスDLL
gfs_base.dll	ハッキングツール検出エンジンインターフェイスDLL
gfs_fact.dll	ハッキングツール検出エンジンインターフェイスDLL
gfs_file.dll	ハッキングツール検出エンジンインターフェイスDLL

gfs_mem.dll	ハッキングツール検出エンジンインターフェイスDLL
gfs_os.dll	ハッキングツール検出エンジンインターフェイスDLL
gfs_proc.dll	ハッキングツール検出エンジンインターフェイスDLL
gfs_util.dll	ハッキングツール検出エンジンインターフェイスDLL
moduler.scd	ハッキングツールパターンエンジンファイル
option.scd	ハッキングツールパターンエンジンファイル

モニタリング関数呼び出し

モニタリングサーバーは、AhnLab_HSMS_Install_Guid および、AhnLab_HSMS_Operator_Guide を参照の上、インストール・操作してください。

詳細は、[4.モニタリングサービス](#) を参照してください。

ライセンスキーの発行

HackShield の API を使用するには Ahnlab,Inc. からライセンスキーの発給を受ける必要があります。

ライセンスキーの取得は以下の手順で行います:

- 1 EhSvc.dllを利用するゲームプログラム実行ファイルの名前、ゲームパブリッシャー、ゲーム開発会社、ゲームタイトル名をAhnlab,Inc.に知らせます。
- 2 固有のゲームコード（4桁の数字）と24桁の文字列で構成されているライセンスキーの発給が完了します。
- 3 発給を受けたゲームコードとライセンスキーの値を初期化関数 `_AhnHS_Initialize` のパラメータとして与えます。

注意

初期化関数 `_AhnHS_Initialize` のパラメータとして正しくない値を与えると、初期化関数を正常に呼び出すことができず、エラー(HS_ERR_INVALID_LICENSE)を返します。

初期化関数呼び出し

プログラミングの準備作業が終わったら、まず始めに初期化関数 `_AhnHS_Initialize` を呼び出します。初期化関数の呼び出しに成功すると HackShield のハッキング遮断機能およびハッキングツール検出機能を実行できるようになります。

初期化関数はゲームクライアントのプログラムのインスタンスやメイン画面を初期化するルーチンから呼び出します。

初期化関数呼び出しのサンプルは次のとおりです。サンプルではすべてのオプションが使えるように初期化関数の最終パラメータとして `AHNHS_CHKOPT_ALL` を使用しています。

注意

以下のCore Option は、自動的に適用されます。

```
AHNHS_CHKOPT_SPEEDHACK,  
AHNHS_CHKOPT_READWRITEPROCESSMEMORY,  
AHNHS_CHKOPT_KDTRACER,  
AHNHS_CHKOPT_OPENPROCESS,  
AHNHS_CHKOPT_AUTOMOUSE,  
AHNHS_CHKOPT_PROCESSSCAN
```

サンプル

```
BOOL HS_Init()  
{  
    int          nRet = 0;  
    TCHAR  szFullPath[MAX_PATH];  
    TCHAR  szMsg[MAX_PATH];  
    DWORD dwOption = 0;  
  
    // ① HackShieldフォルダのEhSvc.dllファイルのパスを指定  
    lstrcat ( szFullPath, _T( "¥¥HackShield¥¥EhSvc.dll" ) );  
  
    // ② _AhnHS_Initialize関数を呼び出す時使うoption flagを定義  
    dwOption = AHNHS_CHKOPT_ALL;  
  
    // ③ _AhnHS_Initialize関数を呼び出してHackShieldサービスを初期化する  
  
    nRet = _AhnHS_Initialize ( szFullPath,  
                                HS_CallbackProc, // callback function  
                                1000,             // game code  
                                "B228F291B7D7FAD361D7A4B7", // License key
```

```

dwOption,    // option flag
AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL );

// ④_AhnHS_Initialize関数の戻り値をチェックしてエラーの処理を行う
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:
        case HS_ERR_NOT_INITIALIZED:
        default:
            wsprintf( szMsg, “ハッキング防止機能に問題が発生しました。(%)”, nRet );
            break;
    }
    MessageBox( NULL, szMsg, szTitle, MB_OK );
    return FALSE;
}
return TRUE;
}

```

注意

初期化関数 `AhnHS_Initialize` は 1 プロセスに対して 1 度だけ呼び出すことができます。初期化関数を重複して呼び出すとエラーが発生することがあります。

EhSvc.dll ファイルが偽造/改ざんされていたり、バージョンが一致しない場合には初期化関数はエラー値 (`HS_ERR_INVALID_FILES`) を返します。

参考

インターフェイスDLL `EhSvc.dll` を呼び出す前の段階では、HackShield の保護機能は動作していません。この状態では、ゲームクライアントのファイルの改ざんを検知・遮断することができません。この脅威を防ぐため、ゲームクライアントのファイルをパッカーなどで暗号化、圧縮することを推奨します。

しかし、実行ファイルのクラッキングから防御するために HackShield はサーバー上にクラック防止のためのインターフェイスを提供します。

一部のユーザーやハッカーたちは、プログラムにアタックするためにOSの下位互換モードでプログラムを実行します。もし利用中のシステムがWindows XP の場合、下位互換

モードではWindows 98/Me または、Windows 2000とみなされて動作します。このような状況では、システムが予期しない状況になることが予想されます。HackShield 初期化モジュールを実行した時に、下位互換モードで動作していることが検出されるとエラー(HS_ERR_COMPATIBILITY_MODE_RUNNING)を返します。

初期化関数を呼び出す際、デフォルトのオプションに加え、追加のオプションを指定することができます。

サービス開始関数呼び出し

HackShield のハッキング遮断およびハッキングツール検出機能を実行するにはサービス開始関数 `_AhnHS_StartService` を次のサンプルのように呼び出します:

サンプル

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

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

    // ② _AhnHS_StartService関数の戻り値をチェックしてエラーの処理を行う。
    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;
}
```

サービス開始関数 `_AhnHS_StartService` は初期化関数 `_AhnHS_Initalize` の呼び出しが正常に完了した後で呼び出す必要があります。

注意

初期化関数を呼び出した直後にサービス開始関数を呼び出すことでゲームクライアントのプログラムをより徹底して保護することができます。時間差が広がるほどハッキングツールがゲームクライアントに侵入する可能性が高まります。

Callback関数の作成

サービス開始関数を呼び出すと現在のサービス状態をリアルタイムでイベントにて通知します。Callback 関数で各イベントを処理するには下記サンプルのように作成します。

サンプル

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

    // ① ケース別エラーメッセージ出力
    switch ( ICode )
    {
        // ② Engine Callback
        case AHNHS_ENGINE_DETECT_GAME_HACK:
            wsprintf( szMsg, “下記のプログラムとゲームは同時に実行できません (%x) ¥n [%s]”, ICode, (LPTSTR)pParam );
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;
        // ③ AutoMacro 検知
        case AHNHS_ACTAPC_DETECT_AUTOMACRO:
            wsprintf(szMsg, _T(“AutoMacroが検知されました (Code = %x)”), ICode);
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;
        // ④ Speed関連
        case AHNHS_ACTAPC_DETECT_SPEEDHACK:
            wsprintf( szMsg, “speed hackが検知されました (%x)”, ICode );
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;
        // ⑤ デバッギング防止
        case AHNHS_ACTAPC_DETECT_KDTRACE:
        case AHNHS_ACTAPC_DETECT_KDTRACE_CHANGED:
            wsprintf( szMsg, “ゲームに対するデバッギングが検知されました(%x)”, ICode );
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;
        // ⑥ HackShield通常動作中
        case AHNHS_ACTAPC_STATUS_HACKSHIELD_RUNNING:
            // HackShieldが通常の動作をしている状態
            // ゲーム開発会社で定義したLogigを実行
            DWORD *dwParam = (DWORD*)pParam;
            break;
        // ⑦ その他のハッキング防止機能異常
        case AHNHS_ACTAPC_DETECT_DRIVERFAILED:
```

```

case AHNHS_ACTAPC_DETECT_HOOKFUNCTION:
case AHNHS_ACTAPC_DETECT_MODULE_CHANGE:
case AHNHS_ACTAPC_DETECT_LMP_FAILED:
case AHNHS_ACTAPC_DETECT_MEM_MODIFY_FROM_LMP:
case AHNHS_AHNHS_ACTAPC_DETECT_ENGINEFAILED:
case AHNHS_ACTAPC_DETECT_CODEMISMATCH:
case AHNHS_ACTAPC_DETECT_ANTIFREESERVER:
case AHNHS_ACTAPC_DETECT_ABNORMAL_HACKSHIELD_STATUS:
    wsprintf( szMsg, “ハッキング防止機能に問題が発生しました。(%%x)”, lCode );
    MessageBox( NULL, szMsg, szTitle, MB_OK );
    break;

return 1;
}

```

注意

AHNHS_ACTAPC_STATUS_HACKSHIELD_RUNNING コールバックの場合、HackShieldの動作状態をチェックする_AhnHS_CheckHackShieldRunningStatus関数を呼び出す時に発生するCallbackです。

このCallbackはHackShieldが動作する時、定期的に発生します。
定期的なCallback処理ができない場合はHackShieldの動作に異常がある可能性があります。Callback発生以後のLogicは開発会社のポリシーに合わせて作成してください。

サービス停止関数呼び出し

ゲームクライアントのプログラムを終了する前に、まずサービス停止関数_AhnHS_StopService を呼び出してハッキング遮断機能およびハッキングツール検出機能を停止します。

ゲームクライアントのプログラムを終了せずに HackShield サービスのみ一時停止する場合には、サービス停止関数 _AhnHS_StopService を呼び出した後でサービス開始関数 _AhnHS_StartService を再度呼び出します。

サンプル

```

BOOL HS_StopService()
{
    int nRet = 0;

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

    if ( nRet != HS_ERR_OK )

```

```

    {
        return FALSE;
    }
    return TRUE;
}

```

サービス終了関数呼び出し

ゲームクライアントのプログラムを完全終了するには、次のサンプルのようにサービス終了関数 `_AhnHS_Uninitialize` を呼び出してプログラムに使われたメモリを開放します。

```

サンプル
BOOL HS_UnInit()
{
    int nRet = 0;

    // ①_AhnHS_Uninitialize関数呼び出してHackShieldサービスを終了する
    nRet = _AhnHS_Uninitialize();

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

```

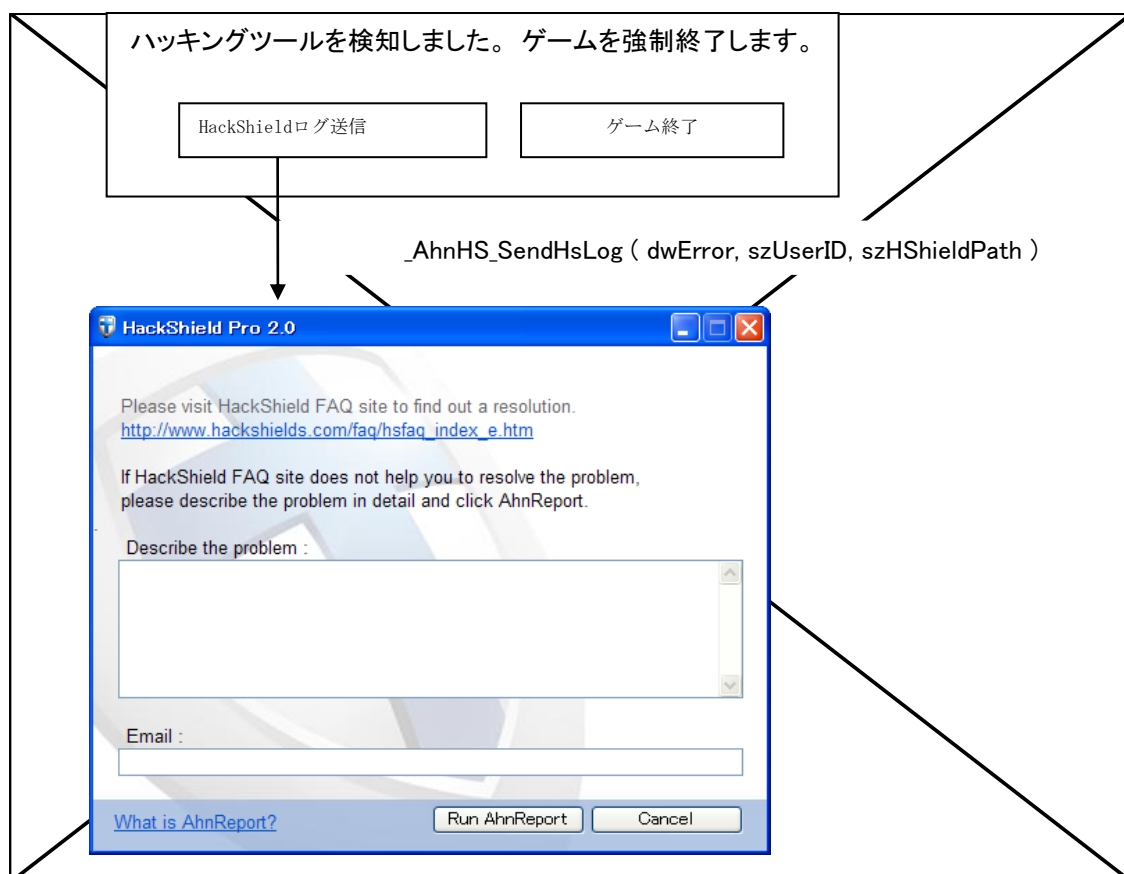
サービス終了関数 `_AhnHS_Uninitialize` が呼び出されると、それ以降ゲームクライアントは Hackshield からの保護を受けることができません。そのため、サービス終了関数はゲームクライアントを終了する最終段階で呼び出す必要があります。

注意

ゲームクライアントのプログラムが正常に終了しないと HackShield のハッキング遮断ドライバをアンロードすることができません。その状態で初期化関数 `_AhnHS_Initialize` を再度呼び出すと、ユーザーのシステムに HackShield のハッキング遮断ドライバが既にロードされているため、エラー (`HS_ERR_INIT_DRV_FAILED`) を返します。その際はシステムを再起動してゲームクライアントのプログラムを最初からもう一度実行する必要があります。

HackShieldログ収集関数呼び出し

この関数は、HackShieldに関連した問題を開発者への通知なしに素早くかつダイレクトに解決するために提供します。ハッキング攻撃に関するログを簡単にAhnLabに送るためには、Callbackコードを処理する際、ゲーム終了ダイアログボックスやメッセージボックスを表示する場合、下記のようにログ送信ボタンを作成して_AhnHS_SendHsLog関数を呼び出してください。



もし、コールバック関数でイベントコード(エラーコード)、ゲームユーザーID、HackShieldのパス情報などを入力して、_AhnHS_SendHsLog 関数を呼び出せば、アンレポートのウィンドウを表示します。エラーの詳細情報とゲームユーザーのメールアドレスを入力した後でアンレポート送信します。

注意

アンレポートのウィンドウはゲームがHackShieldによって終了した場合のみ表示されるように開発してください。他のエラーが発生した場合には「アンレポート実行」の選択ができないメッセージボックスを表示してください。

注意

_AhnHS_SendHsLog の3番目の引数 szHShieldPath は、EhSvc.dll の存在するフォルダのパスを指定してください。その際、EhSvc.dll 除いて指定してください。

Ex) %GameRoot%¥HShield

```
TCHAR szPath[MAX_PATH] = { NULL, }
TCHAR szHShieldPath[MAX_PATH] = { NULL, };
::GetModuleFileName(NULL, szPath, sizeof(szPath)/sizeof(*szPath));
TCHAR* szPos = _tcsrchr(szPath, _T('¥')) + sizeof(_T('¥'));
*szPos = _T('¥0');
_stprintf(szHShieldPath, _T("%s¥HShield"), szPath );
_AhnHS_SendHsLog(dwError, "UserID", szHShieldPath);
```

参考

アンレポートのウィンドウの「エラーの詳細情報」と「メールアドレス」は必須ではありません。入力の必要はありません。入力があった場合もアンラボは、エンドユーザーに直接返信をすることはありません。

参考

アンレポートウィンドウは、Window 7、Windows XP、Windows 2000、Windows Server 2003でサポートしています (Windows 9x ではサポートしていません。)。また、日本語、英語、韓国語、中国語で動作することが可能です

2.4. Application Programming Interface

AhnHS_Initialize

DESCRIPTION

HackShield を初期化してオプションを設定します。プログラムが初期化される時に 1 度だけ呼び出すことができます。他のゲームプログラムで HackShield を使用していたりサービスが正常に終了しなかった場合、エラーが発生することがあります。

SYNTAX

```
EHSVC_API
int __stdcall
_AhnHS_Initialize(
    const char* szFileName
    PFN_AhnHS_Callback pfn_Callback,
    int nGameCode,
    const char* szLicenseKey,
    DWORD dwOption,
    UINT unSHackSensingRatio
);
```

PARAMETERS

Parameter	Value	Description
szFileName		EhSvc.dll のフルパス
Pfn_Callback		Callback 関数へのポインタ
nGameCode	4桁の数字	ゲームごとのユニークなID番号
szLicenseKe	24桁の文字列	ゲームごとのユニークなライセンスキー
dwOption		初期化オプション設定
unSHackSensingRatio		スピードハック検知レベル

OPTIONS

注意

以下のCore Option は、自動的に適用されます。

AHNHS_CHKOPT_SPEEDHACK, AHNHS_CHKOPT_READWRITEPROCESSMEMORY,

**AHNHS_CHKOPT_KDTRACER, , AHNHS_CHKOPT_OPENPROCESS,
AHNHS_CHKOPT_AUTOMOUSE, AHNHS_CHKOPT_PROCESSSCAN**

AHNHS_CHKOPT_SPEEDHACK (Core Option)

スピードハック遮断機能を使用します。ハードウェアコントロールやソフトウェア方式のスピードハックを発見するとスピードハックが動作していないかを Callback 関数によって通知します。スピードハックを診断する検知率レベルの設定によって検知の精度が変わります。SPEEDHACK SENSING RATIO を参照してください。

AHNHS_CHKOPT_READWRITEPROCESSMEMORY (Core Option)

HackShield DLL を使用している現在のプロセスのメモリを外部プロセスから読み取り/書き込みすることができないよう遮断します。

AHNHS_CHKOPT_KDTRACER (Core Option)

カーネルモードデバッガの実行を検知してゲームプロセスに通知します。

AHNHS_CHKOPT_OPENPROCESS (Core Option)

HackShield DLL を使用している現在のプロセスに関する情報を引き出す OpenProcess API 関数の呼び出しを遮断します。

AHNHS_CHKOPT_AUTOMOUSE (Core Option)

オートマウスタ입のプログラムが現在のプロセスに影響を与えないように遮断します。

AHNHS_CHKOPT_PROCESSSCAN (Core Option)

定期的にプロセスリストをスキャンしてハッキングツールが実行していないかをスキャンします。

AHNHS_CHKOPT_ALL

上記で説明したオプションをすべて含むオプションです。

AHNHS_CHKOPT_MESSAGEHOOK

メッセージフッキングを遮断します。

AHNHS_CHKOPT_LOCAL_MEMORY_PROTECTION

指定ファイルのメモリー領域を保護します。

AHNHS_CHKOPT_ANTIFREESERVER

ゲームサーバー以外のサーバーへの接続を許可しません。

AHNHS_USE_LOG_FILE

HackShield の動作に関連したログを記録します。ログファイルは EhSvc.dll と同じディレクトリに hshield.log という名前で生成されます。ログファイルは問題が発生した場合に原因を分析するのに使用します。ログファイルは暗号化されているため、専用のビューアーがない場合は内容の確認ができません。

AHNHS_ALLOW_SVCHOST_OPENPROCESS

NT 系の OS で駆動しているサービスプログラムの SvcHost でゲームプロセスをオープンできるようにします。Windows XP では新たに追加された Windows Audio サービスを使ってサウンドを出力してクライアントプロセスをオープンするため OpenProcess 遮断オプションを使うとサウンドが正常に出力されないことがあります。そのため、Windows XP で DirectMusic または DirectSound からのサウンド出力が正常に動作しない場合はこのオプションを追加します。

AHNHS_ALLOW_LSASS_OPENPROCESS

NT 系の OS で駆動しているサービスプログラムの LSASS.exe でゲームプロセスをオープンできるようにします。ゲームクライアントの内部で直接クレジットカード決済のコントロールする場合に LSASS.exe でゲームプロセスにアクセスするため、このオプションを使ってプロセスのオープンを許可します。

AHNHS_ALLOW_CSRSS_OPENPROCESS

NT 系の OS で駆動しているサービスプログラムの CSRSS.exe でゲームプロセスをオープンできるようにします。ゲームクライアントの内部で直接クレジットカード決済のコントロールする場合に CSRSS.exe でゲームプロセスにアクセスするため、このオプションを使ってプロセスのオープンを許可します。

AHNHS_DONOT_TERMINATE_PROCESS

プロセスリストをスキャンしてハッキングツールを検出すると該当するハッキングツールのプロセスを強制終了することをデフォルトオプションとしています。ハッキングツールのプロセスを強制終了しない場合、このオプションを使用します。AHNHS_CHKOPT_PROCESSSCAN オプションを使う際にはこのオプションを合わせて使

用することを推奨します。

AHNHS_DISPLAY_HACKSHIELD_LOGO

初期化時に HackShield ロゴを画面に表示します。別のスレッドで動作し、2 秒後にロゴは自動的に消えます。HackShield が動作していることを示すために使用します。

参考

ゲーム開発会社の提供するロゴイメージを利用するときは、イメージファイルを `hslogo.jpg` にリネームして `[Game Directory]\¥HShield フォルダー` にコピーしてください。

AHNHS_CHKOPT_PROTECTSCREEN

このオプションはゲーム画面をキャプチャするハッキング攻撃をより効果的に遮断することができます。

注意

このオプションを利用すると他のプログラムや画面キャプチャツールが正常に動作しない可能性があります。このオプションの利用を検討する場合、アンラボまでご相談ください。

AHNHS_ALLOW_SWITCH_WINDOW

フルスクリーン状態で実行するゲームを強制的にウインドウモード化するハッキングツールがあります。そのハッキングツールを検知する機能を使う場合、ゲーム実行中に他のアプリケーションとの画面転換を許可するオプションです。ウインドウモード実行ができないゲーム専用を機能ですのでウインドウモード実行が可能なゲームの場合は使わないでください。

AHNHS_CHKOPT_STANDALONE

通常、HackShieldは多重ロードを許可しているので、ゲームクライアントを複数起動することが可能ですが、このオプションを利用すると同一コードのゲームは1つしか起動できなくなってゲームを多重ロードする場合はエラーコードをリターンします。

AHNHS_CHKOPT_PROTECT_D3DX

DirectXモジュールの VTable をフックするハッキングツールからの遮断機能をアクティブにします。Windows NT 以降の OSで利用可能です (Win9x では利用することができません)。

参考

FPS (First Person Shooting)ゲームなどでは、このオプションを利用することをお勧めします。このタイプのゲームはハッキングの 多くが DirectX ハッキングツールによるものだからです。

AHNHS_CHKOPT_SELF_DESTRUCTION

HackShield がハッキング攻撃を検知し、コールバック関数を呼び出した後になってもゲームを終了できない場合、HackShield が一定時間 (1分間) 経過後にプロセスを終了する機能です。

このオプションは、ハッキング攻撃によりコールバック関数が書き換えて、HackShield の機能が無効になることを防止するための機能です。

このオプションを利用する場合、HackShieldがプロセスを強制的に終了しますので、必ずコールバック関数からHackShieldエラーリターン後の1分以内にゲームプロセスを終了してください。

このオプションを利用する際には、以下の表を参照してコールバック関数の返り値を定め、処理を終了してください。もし、コールバック処理をしなかったり以下にない返り値を利用したりするとHackShield はゲームを異常終了させることがあります。

表 2-3 [AHNHS_CHKOPT_SELF_DESTRUCTION利用時のコールバックコード]

適用オプション	Callback 関数に適用する CallbackCode
必須適用callback code	AHNHS_ACTAPC_DETECT_AUTOMACRO
	AHNHS_ACTAPC_DETECT_SPEEDHACK
	AHNHS_ENGINE_DETECT_GAME_HACK
	AHNHS_ACTAPC_DETECT_ABNORMAL_MEMORY_ACCESS
	AHNHS_ACTAPC_DETECT_ENGINEFAILED
	AHNHS_ACTAPC_DETECT_HOOKFUNCTION
	AHNHS_ACTAPC_DETECT_KDTRACE

	AHNHS_ACTAPC_DETECT_CODEMISMATCH
	AHNHS_ACTAPC_DETECT_ABNORMAL_HACKSHIELD_STATUS
	AHNHS_ACTAPC_DETECT_DRIVERFAILED
AHNHS_CHKOPT_LOCAL_MEMORY_PROTECTION	AHNHS_ACTAPC_DETECT_MEM_MODIFY_FROM_LMP
	AHNHS_ACTAPC_DETECT_LMP_FAILED
AHNHS_CHKOPT_ANITFREESERVER	AHNHS_ACTAPC_DETECT_ANTIFREESERVER
AHNHS_CHKOPT_PROTECTSCREENEX	AHNHS_ACTAPC_DETECT_PROTECTSCREENFAILED

サンプル

```
int __stdcall AhnHS_Callback(long ICode, long IParamSize, void* pParam)
{
    switch ( ICode )
    {
        //Engine Callback
        case AHNHS_ENGINE_DETECT_GAME_HACK:
        case AHNHS_ACTAPC_DETECT_SPEEDHACK:
        case AHNHS_ACTAPC_DETECT_HOOKFUNCTION:
        case AHNHS_ACTAPC_DETECT_KDTRACE:
        case AHNHS_ACTAPC_DETECT_ABNORMAL_MEMORY_ACCESS:
        case AHNHS_ACTAPC_DETECT_ENGINEFAILED:
        case AHNHS_ACTAPC_DETECT_AUTOMACRO:
        case AHNHS_ACTAPC_DETECT_CODEMISMATCH:
        case AHNHS_ACTAPC_DETECT_MEM_MODIFY_FROM_LMP:
        case AHNHS_ACTAPC_DETECT_ANTIFREESERVER:
        case AHNHS_ACTAPC_DETECT_ABNORMAL_HACKSHIELD_STATUS:
            MessageBox ( );
            ExitGame ( );
            break;
    }
    return 1;
}
```

AHNHS_DISPLAY_HACKSHIELD_TRAYICON

このオプションを利用するとシステムトレイにHackShield トレイアイコンを表示することができます。

AHNHS_CHKOPT_DETECT_VIRTUAL_MACHINE

仮想マシン、エミュレーターからゲームを保護します。
このオプションを利用すると仮想マシン、エミュレータ上でゲームの実行を遮断します。
この機能の場合、仮想マシンの機能の変更によって検知Logicが追加及び変更されることがあります。

参考

AHNHS_CHKOPT_DETECT_VIRTUAL_MACHINE オプションは、以下の仮想マシンを検知します。

Virtual PC、VMWare Workstation、Virtual Box、Parallels Workstation
(このオプションを利用しても Virtual Box、Parallels Workstation 上の Windows 9x ゲストOSを検知することはできません。)

AHNHS_CHKOPT_UPDATED_FILE_CHECK

HackShield アップデートが終了した後、更新したファイルでHackShieldの動作チェックを行います。

注意

この機能を利用するには、以下の制約があります。:

- HackShield アップデート関連モジュール V3Hunt.dll ファイルが HackShield を実行するフォルダに存在する必要があります。
- HackShield アップデート関連ファイル HSUpdate.env ファイルがHackShield を実行するフォルダに存在する必要があります。
- HSUpdate.env ファイルに指定できるアップデートサーバは 'HTTP' のみサポートします。
(もし HSUpdate.env ファイルに FTP などのアドレスを記入した場合、エラーが発生し、ゲームクライアントにコールバックします。)

このオプションを利用するには、以下の設定を推奨します。:

- _AhnHS_HSUpdateEx() を使用してアップデートを実行します。
(関数呼び出しの際に、“Game code” を引数に指定する必要があります。)
 - “Game code” を HSUpdate.env ファイルに保存するためには、HSUpSetEnv.exe ツールを利用します。
-

SPEEDHACK SENSING RATIO

AHNHS_SPEEDHACK_SENSING_RATIO_HIGHEST

最も敏感なレベルです。検出値は36.5% と -3.0% です。
基準から 1.0718 倍以上・0.8409 倍以下のスピードに変更された場合スピードハックプログラムと判定します。
(ハードウェアによるスローダウンは検出しません。)

AHNHS_SPEEDHACK_SENSING_RATIO_HIGH

敏感なレベルです。検出値は30.5% と -9.5% です。
基準から 1.1487 倍以上・0.8123 倍以下のスピードに変更された場合、スピードハックプログラムと判定しす。

AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL

通常のレベルです。検出値は26.0% と -12.5% です。

基準から 1.1892 倍以上・ 0.7846 倍以下のスピードに変更された場合、スピードハックプログラムと判定しす。

AHNHS_SPEEDHACK_SENSING_RATIO_LOW

低精度なレベルです。検出値は22.5% と 15.5% です。

基準から 1.2311 倍 以上・ 0.7579 倍以下のスピードに変更された場合、スピードハックプログラムと判定しす。

AHNHS_SPEEDHACK_SENSING_RATIO_LOWEST

最も低精度なレベルです。検出値は17.5% と 18.5% です。

基準から 1.2746 倍以上 ・ 0.7320 倍以下のスピードに変更された場合、スピードハックプログラムと判定しす。

RETURN VALUE

HS_ERR_OK (Value = 0x0000)

内容: 関数の呼び出しに成功した時に返される値です。

HS_ERR_INVALID_PARAM(Value = 0x0002)

内容: パラメータが不正です。

原因: Callback 関数のポインタが正しく設定されていなかったり、ライセンスキーの値が NULL である場合に発生するエラーです。

対処方法: 開発過程においてのみ発生するエラーであるため、特に処理する必要はありません。

HS_ERR_NOT_INITIALIZED(Value = 0x0003)

内容: 初期化に失敗しました。

原因: HackShield が利用するシステムライブラリの読み込みに失敗すると発生します。

対処方法: この問題が発生した場合、アンラボにお問い合わせください。

HS_ERR_COMPATIBILITY_MODE_RUNNING (Value = 0x0004)

内容: ゲームが互換性モードで起動されました。

原因: ゲームクライアントが Windows XP 系列で提供している互換性モード (Compatibility Mode) で実行されると発生します。

対処方法: 互換性モードでゲームを実行した場合、悪意の目的を持ったユーザーと判断され、プログラムを強制終了して再起動する必要があります。

HS_ERR_EXCEPTION_RAISED (Value = 0x007)

内容: 例外が発生しました。

原因: HackShield の動作中に例外(Exception)が発生したことが原因です。

対処方法: アンラボに HShield.log ファイルと AhnReport を送ってご相談ください。

HS_ERR_INVALID_LICENSE (Value = 0x100)

内容: ライセンスキーが不正です。

原因: 初期化関数のパラメータとして設定したゲームコードやライセンスキーの値が実際の値と一致しない時に発生するエラーです。

対処方法: ライセンスキーを確認してください。

HS_ERR_INVALID_FILES (Value = 0x101)

内容: HackShield ファイルが不正です。

原因: インターフェース DLL ファイル (EhSvc.dll) が偽造/改ざんされたり、バージョンが一致しない場合に発生するエラーです。インターフェース DLL ファイルが偽造/改ざんされるとハッキング遮断機能が無力化することがあります。

対処方法: HackShield ファイルのバージョンが前のバージョンになっていないか確認してください。また、HackShield フォルダのファイルが存在するかを確認してください。

HS_ERR_INIT_DRV_FAILED (Value = 0x102)

内容: HackShield ドライバの起動を失敗しました。

原因: ハッキング遮断機能のドライバの初期化に失敗した時に発生するエラーです。ドライバの初期化に失敗するとハッキング遮断機能が正常に動作できないため、プログラムを強制終了して再起動する必要があります。

対処方法: システムフォルダにある EagleNT.sys ファイルの権限を確認してください。また、EagleNT.sys ファイルが動作しているかアンロードされたかを確認してください。

HS_ERR_ALREADY_INITIALIZED (Value = 0x104)

内容: HackShield は既に初期化済みです。

原因: _AhnHS_Initialize を利用して既に初期化されている場合に発生します。

対処方法: AhnHS_Initialize はプログラムの初期化時点で 1 度だけ呼び出すことができ、重複して呼び出すことはできません。重複して呼び出していないか確認してください。

HS_ERR_DEBUGGER_DETECT (Value = 0x105)

内容: デバッガを検出しました。

原因: システムでデバッガーが実行している場合に発生します。。

対処方法: 通常開発工程中の場合、リリースバージョンではないDevバージョン(開発バージョン)を利用してデバッグを行ってください。(Remark 参照)

ユーザーがゲームをプレイ中にこのエラーが発生した場合、ハッキング攻撃の可能性があります。そのため、ゲームをただちに終了してゲームを起動しなおす必要があります。

デバッガと無関係にこのエラーが発生する場合には、アンラボにお問い合わせください。

HS_ERR_NEED_ADMIN_RIGHTS (Value = 0x107)

内容: システム管理者権限が必要です。

原因: ゲームクライアントが一般ユーザー権限のアカウントで実行されると発生します。

対処方法: ゲームクライアントを一般ユーザー権限で動作させるには、ゲームを動作させる前に HackShield のシャドウアカウントを作成します。詳細は、‘8.2.ユーザー権限実行サポート機能’を参照してください。

HS_ERR_MODULE_INIT_FAILED (Value = 0x108)

内容: HackShield モジュールの初期化に失敗しました

原因: HackShield の初期化中に問題があると発生します。

対処方法: アンラボに HShield.log ファイルと AhnReport を送ってご相談ください。

HS_ERR_UNKNOWN (Value = 0x001)

内容: 不明なエラーが発生しました。

原因: 関数の例外の発生や関数の構造的な問題の可能性が高いです。

対処方法: アンラボに HShield.log ファイルと AhnReport を送ってご相談ください。

REMARKS

開発プロセス、テストプロセスでゲームクライアントをデバッグするためには
開発バージョンのHShield.lib、Ehvsc.dll、hshield.dat を使用してください。
(Ehvsc.dll、hshield.dat をリリース版と混用したらサーバー連動時に問題が発生する可能性があります。)

– Path: ¥Developer¥

開発バージョンのモジュールを利用したら以下のロゴが表示されます。

(クリックするとなくなります。)



AhnHS_Callback

DESCRIPTION

ハッキング遮断やハッキングツール検出をした結果に関する情報を伝える Callback 関数であり、次のようなイベントを伝えます。:

ハッキング遮断機能に関連したイベント
ハッキングツール検出機能に関連したイベント

STRUCTURE

```
int __stdcall AhnHS_Callback(  
    long ICode,  
    long IParamSize,  
    void* pParam  
);
```

PARAMETERS

Parameter	Value	Description
ICode	Long	イベントコード
IParamSize	Long	イベントパラメーターのサイズ
pParam	Void*	イベントパラメータ

EVENTS

AHNHS_ENGINE_DETECT_GAME_HACK

システム上で実行されているゲーム専用ハッキングツールを検出した場合、Callback 関数に伝えるイベントです。受け取ったハッキングツール情報を使ってゲーム開発会社はハッキングツールの実行ファイルを強制的に削除するなどといった対応をとることができます。

pParam: 検出されたゲーム専用ハッキングツールの実行ファイル名(ファイルパスを含む)
IParamSize: 検出されたハッキングツールの実行ファイル名の長さ

AHNHS_ENGINE_DETECT_WINDOWED_HACK

ハッキングツールによりフルスクリーンモードからウインドウモードに切り替えられたことを検知しました。ウインドウモードで動作しないゲームにのみ適用されます。ウインドウモ

ードをサポートするゲームやフルスクリーンモードとウインドウモードの両方に対応するゲームの場合利用できません。

pParam: NULL

IPParamSize: 0

AHNHS_ACTAPC_DETECT_ALREADYHOOKED

ハッキング遮断機能が実行中である時、ハッキング遮断機能で保護する API 関数が既にフッキングされていることを知らせるイベントです。ハッキングツールではない通常プログラムでも機能を実行するためにフッキングすることがあるため、ハッキングツールかどうかの判断はゲーム開発会社の内部規則によって決定する必要があります。

pParam: ACTAPCPARAM_DETECT_HOOKFUNCTION*

IPParamSize: ACTAPCPARAM_DETECT_HOOKFUNCTIONStructure 構造体の長さ

AHNHS_ACTAPC_DETECT_HOOKFUNCTION

ハッキング遮断機能が実行中である時、Win32 関数または、保護対象の関数がフックされると、このイベントが発生します。このイベントが発生した場合は、ハッキングツールによる可能性が高いので強制的にゲームプログラムを終了する必要があります。

pParam: ACTAPCPARAM_DETECT_HOOKFUNCTION*

IPParamSize: ACTAPCPARAM_DETECT_HOOKFUNCTION 構造体の長さ

```
struct _ACTAPCPARAM_DETECT_HOOKFUNCTION
{
    char szFunctionName[128];
    char szModuleName[128];
}
ACTAPCPARAM_DETECT_HOOKFUNCTION,
*PACTAPCPARAM_DETECT_HOOKFUNCTION;
```

szFunctionName: フックされた関数の名前

szModuleName: フックされたモジュール名

AHNHS_ACTAPC_DETECT_AUTOMOUSE

オートマウスタイプのプログラムを使ってキーボードやマウスを操作し、自動的にデータを入力する場合に発生するイベントです。

pParam: ACTAPCPARAM_DETECT_AUTOMOUSE

IPParamSize: ACTAPCPARAM_DETECT_AUTOMOUSE 構造体の長さ

```
typedef struct
{
    BYTE byDetectType;
    DWORD dwPID;
```

```

        CHAR    szProcessName[16+1];
        CHAR    szAPIName[128];
    } ACTAPCPARAM_DETECT_AUTOMOUSE, *PACTAPCPARAM_DETECT_AUTOMOUSE;

```

byDetectType の値や意味は以下のとおりです。dwPID、szProcessName、szAPIName は現在使用していません。

EAGLE_AUTOMOUSE_APCTYPE_SHAREDMEMORY_ALTERATION

(3): オートマウス遮断機能をするライブラリの内部データを変更します。HackShield がハッキングを遮断するために使用する内部情報をハッキングプログラムが修正した場合、遮断機能が動作できないため、ゲームプログラムを強制終了する必要があります。

EAGLE_AUTOMOUSE_APCTYPE_API_CALLED: キーボード、マウス関連 API 呼び出し

EAGLE_AUTOMOUSE_APCTYPE_API_ALTERATION: API フック操作

AHNHS_ACTAPC_DETECT_AUTOMACRO

オートマウスタイプのプログラムを使ってキーボードやマウスを操作し、自動的にデータを入力する場合に発生するイベントです。正常ではない入力値が繰り返される現象であり、ゲームプログラムを強制終了する必要があります。

pParam: NULL

IParamSize: 0

AHNHS_ACTAPC_DETECT_DRIVERFAILED

ハッキング遮断ドライバがシステムにロードされない場合に発生するイベントです。ハッキング遮断ドライバがシステムにロードされないとハッキング遮断機能が正常に動作しないため、直ちにゲームプログラムを強制終了します。またドライバが正常に取り外されなかった場合にはシステムが不安定になることがあるためシステムを再起動します。

pParam: NULL

IParamSize: 0

AHNHS_ACTAPC_DETECT_SPEEDHACK

システム時間の変化する速度が正常ではない場合に発生するイベントです。ハードウェアまたはソフトウェア方式のスピードハックが実行中である可能性が非常に高いため、時間に精密なゲームプログラムを強制終了します。ハードウェア方式のスピードハックは Windows のバージョンによっては OS でも遮断します。

pParam: (double *) 最近 5 秒間の時間データ情報

IParamSize: pParam で伝えられた時間データの個数

AHNHS_ACTAPC_DETECT_KDTRACE

カーネルレベルあるいはアプリケーションレベルのデバッガによってデバッグトレースが

発生したことを伝えるイベントです。ハッカーのような悪意のユーザーがゲームプログラムに対するデバッグ作業を進行中である可能性が高いため、ゲームプログラムを中止することを推奨します。

pParam: NULL

IParamSize: 0

AHNHS_ACTAPC_DETECT_ABNORMAL_MEMORY_ACCESS

許可されていないプロセスがゲームプロセスのメモリにアクセスする時に発生するイベントです。メモリアccessを試みたプロセスのパスと名前がわかるため、どんなプロセスがアクセスしたのか確認できます。

pParam: 検出されたゲーム専用ハッキングツールの実行ファイル名(ファイルパスを含む)

IParamSize: 検出されたハッキングツールの実行ファイル名の長さ

AHNHS_ACTAPC_DETECT_KDTRACE_CHANGED

デバッグトレース遮断ルーチンが変更された場合に発生するイベントです。このイベントは、ゲームプログラムに対するデバッグトレース遮断ルーチンが実行されている間カーネルレベルのデバッグプログラムがアクティブであることを意味するため、ゲームプログラムを強制終了する必要があります。

pParam: NULL

IParamSize: 0

AHNHS_ACTAPC_DETECT_MODULE_CHANGE

HackShield モジュールに対する偽造/改ざんが検知されたことを伝えるイベントです。HackShield が正常に動作しないことがあるため、ゲームプログラムを強制終了する必要があります。

pParam: NULL

IParamSize: 0

AHNHS_ACTAPC_DETECT_ENGINEFAILED

HackShield のヒューリスティックエンジン (3n.mhe) ファイルが削除されたり、正常にロードされなかったことを伝えるイベントです。HackShield の ProcessScan 機能が正常に動作しないことがあるため、ゲームを強制終了する必要があります。

pParam: NULL

IParamSize: 0

AHNHS_ACTAPC_DETECT_CODEMISMATCH

HackShield のインターフェースモジュールである Ehsvc.dll モジュールのコード領域が操作されたことを伝えるイベントです。HackShield の機能が正常に動作しないことがあるため、ゲームを強制終了する必要があります。

pParam: NULL

IPParamSize: 0

AHNHS_ACTAPC_DETECT_MEM_MODIFY_FROM_LMP

HackShieldが保護しているファイルのメモリ領域が改ざんされたことを伝えるイベントです。このイベントが発生するとモジュール改ざんにより正常に動作しないことがあるため、ゲームを強制終了する必要があります。

pParam: 操作されたモジュール名(改ざんされたページの先頭アドレス)

IPParamSize: 0

AHNHS_ACTAPC_DETECT_LMP_FAILED

ハッキングまたは、システムの問題により LMP 機能が正常に動作しない場合に発生します。システム、ゲームを再起動する必要があります。

IPParamSize: 0

AHNHS_ACTAPC_DETECT_ANTIFREESERVER

このイベントはゲームプログラムが異常なアドレスのサーバーへアクセスしようとするが発生します。この問題が発生した場合、ゲームを強制的に終了する必要があります。pParam にはアドレスが設定されていますが内部使用に留め、ゲームユーザーに公開しないことをおすすめします。

pParam: 接続を試みた IP アドレス

IPParamSize: 0

AHNHS_ACTAPC_DETECT_ABNORMAL_HACKSHIELD_STATUS

このイベントは、HackShield が正常に動作していない場合に発生します。ゲームを強制的に終了する必要があります。

pParam: NULL

IParamSize: 0

AHNHS_ACTAPC_DETECT_PROTECTSCREENFAILED

仮想デスクトップを利用してゲームを保護する機能が攻撃されて動作できない状態になると発生します。このイベントが発生したら仮想デスクトップ機能が動作させることができないので、仮想デスクトップを利用するゲームプログラムの場合はゲームを強制終了する必要があります。

pParam: NULL

IParamSize: 0

AHNHS_ACTAPC_STATUS_HACKSHIELD_RUNNING

HsckShield が正常に動作していることを示すコールバックであり、_AhnHS_CheckHackShieldRunningStatus 関数を呼び出すことでプロセスが始まります。HackShield が正常に動作している場合は、25秒毎にこのイベントが発生します。これは、_AhnHS_StopService 関数と_AhnHS_Uninitialize 関数を呼び出すまで繰り返されます。そのため、このイベントが定期的に発生しない状況は、HackShield が正常に動作しなくなったと判断できます。

pParam: (DWORD *) HackShieldの動作状態値(enum HS_RUNNING_STATUSに定義された値の中で一つ)

IParamSize: pParamのサイズ

```
enum HS_RUNNING_STATUS {  
    HS_RUNNING_STATUS_CHECK_MONITORING_THREAD = 1,  
};
```

注意

AHNHS_ACTAPC_STATUS_HACKSHIELD_RUNNING Callbackは25秒毎に内部的に呼び出されます。システムによっては、25秒のサイクルは、変更されることがあります。

[_AhnHS_StartService](#)

DESCRIPTION

ハッキングツール検出機能とハッキング遮断機能を動作させます。_AhnHS_Initialize 関数を呼び出す前に呼ぶ出すことはできず、重複して呼び出すこともできません。_AhnHS_StopService 関数を呼び出してサービスを停止した場合にはこの関数を再度呼び出してサービスを再開することができます。

SYNTAX

```
EHSVC_API  
int _stdcall  
_AhnHS_StartService( );
```

PARAMETERS

無し.

RETURN VALUE

HS_ERR_OK (Value = 0x000)

内容: 関数の呼び出しに成功した時に返される値です。

HS_ERR_NOT_INITIALIZED (Value = 0x003)

内容: HackShield が初期化されていません。

原因: _AhnHS_Initialize 関数を呼び出さなかったり関数の呼び出しに失敗して、HackShield を初期化していない状態でこの関数を呼び出した場合に発生します。

対処方法: このエラーは開発過程においてのみ発生するエラーであるため、特に処理する必要はありません。

HS_ERR_START_ENGINE_FAILED (Value = 0x200).

内容: エンジンの読み込みに失敗しました。

原因: 初期化関数を呼び出す時にハッキングツールのプロセス検知オプション (AHNHS_CHKOPT_PROCESSSCAN) を設定したにもかかわらず、ハッキングツールのパターンエンジンの初期化に失敗した場合に発生するエラーです。また、ハッキングツ-

ルのパターンエンジン関連ファイルがきちんとインストールされていなかったり、パターンエンジンの関連ファイル（v3pro32s.dll、v3warps.v3d、v3warps.v3d）が HackShield フォルダにきちんとインストールされていない場合に発生します。

対処方法: このエラーが発生した場合、プログラムを強制終了して再起動または再インストールする必要があります。

HS_ERR_ALREADY_SERVICE_RUNNING (Value = 0x201)

内容: HackShield サービスが既に実行中です。

原因: AhnHS_StartService 関数を既に呼び出した状態で再度呼び出した場合に発生するエラーです。

対処方法: この関数を再度呼び出すには _AhnHS_StopService を呼び出してサービスを終了する必要があります。このエラーは開発過程においてのみ発生するエラーであるため、特に処理する必要はありません。

HS_ERR_DRV_FILE_CREATE_FAILED (Value = 0x202)

内容: HackShield ドライバファイルの作成に失敗しました。

原因: ハッキング遮断のためのドライバファイル生成に失敗した場合に発生するエラーです。HackShield プログラムの内部にはハッキング遮断処理をするドライバがあり、ゲーム開始時にこれを生成してロードします。ゲーム開始時、ドライバファイルの生成に問題が起きた場合にこのエラーが発生します。

対処方法: ログインしているセッションでシステムフォルダに書き込み権限があるか確認してください。また、システムフォルダにインストールされる HackShield ドライバファイルの EagleNT.sys ファイルが読み込み専用になっていたりアクセスできない状態になっていたりしたらEagleNT.sys ファイルを削除した後、HackShieldを再起動してください。

HS_ERR_REG_DRV_FILE_FAILED (Value = 0x203)

内容: HackShield ドライバファイルの登録に失敗しました。

原因: ハッキング遮断処理をするドライバファイルをシステムへ登録するのに失敗した時に発生するエラーです。

対処方法: このエラーが発生すると HackShield が正常に動作できないため、プログラムを強制終了して再起動あるいは再インストールします。

HS_ERR_START_DRV_FAILED (Value = 0x204)

内容: HackShield ドライバの実行に失敗しました。

原因: さまざまな原因によってゲームクライアントが正常に終了されなかったために `_AhnHS_StopService` または `_AhnHS_Uninitialize` 関数を呼び出すことができず、それまでロードされていたドライバが停止したり、ロードすべきではないドライバがロードされている状態でゲームを再実行すると発生するエラーです。このエラーが発生するとシステムが不安定になったり、システムに問題が起きることがあるため、プログラムを終了してシステムを再起動する必要があります。

対処方法: ログインしているセッションでシステムフォルダに書き込み権限があるか確認してください。また、システムフォルダにインストールされる HackShield ドライバファイルの `EagleNT.sys` ファイルが読み込み専用になっていたりアクセスできない状態になっていたりしたら `EagleNT.sys` ファイルを削除した後、HackShieldを再起動してください。

HS_ERR_ALREADY_GAME_STARTED (Value = 0x206)

内容: ゲームが既に起動しています。

原因: このエラーは初期化時に HackShield のスタンドアロンオプション (`AHNHS_CHKOPT_STANDALONE`) を適用しているときに発生する場合があります。このエラーが発生した時、既に同一のゲームコードが実行されています。

対処方法: もし HackShield の初期化オプションにスタンドアロンオプション (`AHNHS_CHKOPT_STANDALONE`) を指定している場合、ゲームのプロセスは1つしか起動できません。このエラーが発生した場合には、プログラムを終了させる必要があります。

HS_ERR_VIRTUAL_MACHINE_DETECT (Value = 0x207)

内容: ゲームが仮想マシンまたは、エミュレータ上で起動しています。

原因: 初期化関数を呼び出す時、仮想マシンからの実行を防止するオプション (`AHNHS_CHKOPT_DETECT_VIRTUAL_MACHINE`) を適用している場合、仮想マシン上でゲームを起動すると、発生します。

対処方法: `AHNHS_CHKOPT_DETECT_VIRTUAL_MACHINE` オプションを適用している場合、HackShield によって仮想マシン上での実行されることからゲームを保護します。

AhnHS StopService

DESCRIPTION

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。

SYNTAX

```
EHSVC_API  
int __stdcall  
_AhnHS_StopService( );
```

PARAMETERS

無し.

RETURN VALUE

HS_ERR_OK (Value = 0x0000)

内容: 関数の呼び出しに成功した時に返される値です。

HS_ERR_NOT_INITIALIZED (Value = 0x003)

内容: HackShield が初期化されていません。

原因: _AhnHS_Initialize 関数を呼び出さなかったり関数の呼び出しに失敗して、HackShield を初期化していない状態でこの関数を呼び出した場合に発生するエラーです。

対処方法: このエラーは開発過程においてのみ発生するものであるため特に処理する必要はありません。

HS_ERR_SERVICE_NOT_RUNNING (Value = 0x301)

内容: HackShield サービスが起動していません。

原因: _AhnHS_ StartService関数を呼び出してHackShield を起動していない状態でこの関数を呼び出した場合に発生するエラーです。

対処方法: このエラーは開発過程でのみ発生するエラーであるため、特に処理する必要はありません。

_AhnHS_Uninitialize

DESCRIPTION

システムの内部で使用されていたメモリを開放して変数の終了処理をします。

SYNTAX

```
EHSVC_API  
int __stdcall  
_AhnHS_Uninitialize ( );
```

PARAMETERS

無し.

RETURN VALUE

HS_ERR_OK (Value = 0x0000)

内容: 関数の呼び出しに成功した時に返される値です。

HS_ERR_SERVICE_STILL_RUNNING (Value = 0x302)

内容: HackShield が既に実行しています。

原因: _AhnHS_StopService 関数を呼び出して HackShield が終了していない状態でこの関数を呼び出すと発生するエラーです。

対処方法: このエラーは開発過程においてのみ発生するものであるため特に処理する必要はありません。

HS_ERR_NOT_INITIALIZED (Value = 0x003)

内容: HackShield が初期化されていません。

原因: _AhnHS_Initialize 関数を呼び出さなかったり関数の呼び出しに失敗して、HackShield を初期化していない状態でこの関数を呼び出した場合に発生するエラーです。

対処方法: このエラーは開発過程においてのみ発生するものであるため特に処理する必要はありません。

AhnHS_PauseService

DESCRIPTION

メッセージフッキング防御機能のキーボード防御機能を一時中断します。この操作を行うことで、ゲーム実行中、Microsoft Internet Explorer のウェブページ形式で決済などのためにユーザーがデータを入力する際のユーザー入力が有効となります。キーボード防御機能を一時中止した後は `_AhnHS_ResumeService` を使用して機能を再度アクティブにさせる必要があります。

SYNTAX

```
EHSVC_API
int __stdcall
_AhnHS_PauseService (
    DWORD dwPauseOption
);
```

PARAMETERS

Parameter	Value	Description
<code>dwPauseOption</code>	DWORD	現在は <code>AHNHS_CHKOPT_MESSAGEHOOK</code> オプションのみ使用可能です。それ以外のオプションが伝わると <code>HS_ERR_INVALID_PARAM</code> エラーを返します。

RETURN VALUE

HS_ERR_OK (Value = 0x000)

内容: 関数の呼び出しに成功した時に返される値です。

HS_ERR_NOT_INITIALIZED (Value = 0x003)

内容: HackShield が初期化されていません。

原因: `_AhnHS_Initialize` 関数を呼び出さなかったり関数の呼び出しに失敗して、HackShield を初期化していない状態でこの関数を呼び出した場合に発生するエラーです。

対処方法: このエラーは開発過程においてのみ発生するものであるため特に処理する必要はありません。

HS_ERR_SERVICE_NOT_RUNNING (Value = 0x301)

内容: HackShield サービスが起動していません。

原因: _AhnHS_StartService関数を呼び出し、HackShield を起動していない状態でこの関数を呼び出した場合に発生するエラーです。

対処方法: このエラーは開発過程においてのみ発生するものであるため特に処理する必要はありません。

HS_ERR_INVALID_PARAM (Value = 0x002)

内容: 引数が間違っています。

原因: dwPauseOption が AHNHS_CHKOPT_MESSAGEHOOK でない場合に発生。

AhnHS_ResumeService

DESCRIPTION

AhnHs_Pause service を呼び出して一時中断していた HackShield 機能を再度アクティブにします。メッセージフッキング防御機能のキーボード防御についてのみ処理します。この機能の必要性については「_AhnHS_PauseService」の説明をご参照ください。

SYNTAX

```
EHSVC_API
int __stdcall
_AhnHS_ResumeService (
    DWORD dwResumeOption
);
```

PARAMETERS

Parameter	Value	Description
dwResumeOption	DWORD	現在は AHNHS_CHKOPT_MESSAGEHOOK オプションのみ使用可能です。それ以外のオプションが伝わると HS_ERR_INVALID_PARAM エラーを返します。

RETURN VALUE

HS_ERR_OK (Value = 0x000)

内容: 関数の呼び出しに成功した時に返される値です。

HS_ERR_NOT_INITIALIZED (Value = 0x003)

内容: HackShield が初期化されていません。

原因: _AhnHS_Initialize 関数を呼び出さなかったり関数の呼び出しに失敗して、HackShield を初期化していない状態でこの関数を呼び出した場合に発生するエラーです。

対処方法: このエラーは開発過程においてのみ発生するものであるため特に処理する必要はありません。

HS_ERR_SERVICE_NOT_RUNNING (Value = 0x301)

内容: HackShield サービスが起動していません。

原因: _AhnHS_StartService関数を呼び出し、HackShield を起動していない状態でこの関数を呼び出した場合に発生するエラーです。

対処方法: このエラーは開発過程においてのみ発生するものであるため特に処理する必要はありません。

HS_ERR_INVALID_PARAM (Value = 0x002)

内容: 引数が間違っています。

原因: dwPauseOption が AHNHS_CHKOPT_MESSAGEHOOK でない場合に発生します。

AhnHS_CheckHackShieldRunningStatus

DESCRIPTION

HackShieldの動作状態を定期的に確認してゲームにコールバックする機能を開始します。
ゲーム上のHackShieldの動作に問題があるかどうかを確認するための関数です。
HackShield 起動後(_AhnHS_Initialize, _AhnHS_StartService関数呼び出し)から
終了(_AhnHS_StopService, _AhnHS_UnInitialize関数呼び出し)前までに動作します。

SYNTAX

```
EHSVC_API  
int __stdcall  
_AhnHS_CheckHackShieldRunningStatus ();
```

PARAMETERS

無し

RETURN VALUE

HS_ERR_OK (Value = 0x000)

内容: 関数の呼び出しに成功した時に返される値です。

原因: Normal

HS_ERR_OK がリターンされたらHackShield の操作ステータスが定期的にチェックされ、
ゲームにコールバックします。

HS_ERR_NOT_INITIALIZED (Value = 0x003)

内容: 初期化に失敗しました。

原因: _AhnHS_StartService 関数を呼び出して HackShield を開始していない場合に
_AhnHS_CheckHackShieldRunningStatus関数を呼び出したら発生します。
このエラーが発生した場合、HackShield で利用するシステムライブラリが正しく読み込まれていません。

対処方法: HackShield が起動していないか、ハッキング攻撃に対して正常に対処できていません。

HS_ERR_INVALID_FILES (Value = 0x101)

内容: HackShield が初期化されていません。

原因: _AhnHS_Initialize 関数を呼び出さなかったり関数の呼び出しに失敗して、HackShield を初期化していない状態でこの関数を呼び出した場合に発生するエラーです。

対処方法: HackShield が起動していないか、ハッキング攻撃に対して正常に対処できていません。

参考

この関数を利用すると、AHNHS_ACTAPC_STATUS_HACKSHIELD_RUNNING ステータスが定期的にコールバックされます。このコールバックは、HackShieldの動作状態、(通常、ハッキング検出)を通知するので、ゲームのセキュリティを強化するために使用することができます。

AhnHS_SendHsLog

DESCRIPTION

HackShield のログをアンラボに送信したい場合、HackShieldのイベントコードとゲームユーザーID、HackShield のパスを設定して_AhnHS_SendHsLog 関数を呼び出すとログを送信できます。

この機能を利用したい場合は、ユーザーの選択でログを送ったりすることができるようにLog送信ボタン、終了ボタンがあるコールバック関数のゲーム終了ダイアログを用意してください。

その後、保存ボタンを押した場合に_AhnHS_SendHsLog関数を呼び出した後、ゲームが終了させる必要があります。

SYNTAX

```
void
_stdcall
_AhnHS_SendHsLog (
    IN DWORD dwError,
    IN const char* szUserID
    IN const char* szHShieldPath

);
```

PARAMETERS

Parameter	Value	Description
dwError	DWORD	HackShieldコールバック関数の最初の引数で渡されるイベントコード(lCode)を入力してください。
szUserID	const char *	ゲームユーザーID
szHShieldPath	const char *	EhSvc.dll があるフォルダのフルパス。 (ファイル名は除外)。

RETURN VALUE

なし

3. HackShield アップデート機能

3.1. 概要

HackShield アップデート機能は、既存にHackShieldのパッチを行う時ゲームのパッチに合わせてビルドしてゲームのラウンチャを利用してHackShieldのアップデートを行った方法を改善するために開発しました。HackShield専用のアップデートを利用して簡単にHackShield モジュールにパッチを行ってモジュールの更新ができるようになって迅速なハッキングツール対応ができます。

機能

HackShield モジュールアップデート機能

HackShield アップデートは、HackShield のパッチが必要な時に簡単にモジュールを更新する目的で提供しています。

エンジンアップデート

ハッキングツールの出現に素早く対応するため、Signature, Heuristicエンジンを更新することができます。

HackShield アップデート ON/OFF 機能

HackShield アップデート機能利用途中にHackShieldアップデート機能を使いたくない場合、パッチセットの設定によりオン/オフを切り替えることができます。

Splash image feature

HackShield アップデートを実行中の画面に、開発者が定義した画像を表示することが可能です。

特徴

インターフェイス関数(API)の提供

HSUpChk.libから提供しているAPIを利用してゲームラウンチャや実行ファイルから最初に呼び出されHackShield のモジュール・エンジンを更新することができますようにします。

アップデート環境ファイル作成プログラムの提供

アップデートサーバー環境ファイルを提供してアップデートサーバーのURL設定ができます。(複数のサーバー、複数のURLも対応可能です。)

テストプログラムの提供

HSUpChk.libから提供しているAPIを利用したテストプログラムAmazon.exeを提供します。Amazon.exeは、既存のHackShield テストとHackShieldアップデート機能のテストを提供します。

システムアーキテクチャ

HackShieldアップデートはHSUpChk.libを提供してクライアントからAPIを呼び出してアップデートを行います。HackShieldアップデートのアーキテクチャと動作原理は以下のとおり:

HSUpChk.lib (HackShield アップデートライブラリ)

クライアントから利用されます。アップデート関数が呼び出されると HSUpdate.exe が起動して HackShield モジュールを更新します。

HSUpdate.env (アップデート環境ファイル)

HSUpSetEnv.exeツールによって作成されます。アップデートサーバーの設定情報が含まれます。。

HSUpSetEnv.exe (アップデート環境ファイル作成プログラム)

HSUpdate.env を作成・更新するツールです。開発者はアップデートサーバー関連の設定を変更できます。

noupdate.ui (アップデート OFF 設定ファイル)

noupdate.ui は、一時的にアップデート機能を OFF にするファイルです。

noupdate.ui ファイルをアップデートサーバーのパッチセットの下(ahn.ui と autoaup.exe と同じ場所)に配置すると HackShield アップデート機能は無効になります。

noupdate.ui ファイルをアップデートサーバーのパッチセットの下(ahn.ui と autoaup.exe と同じ場所)から削除すると HackShield アップデートは有効になります。

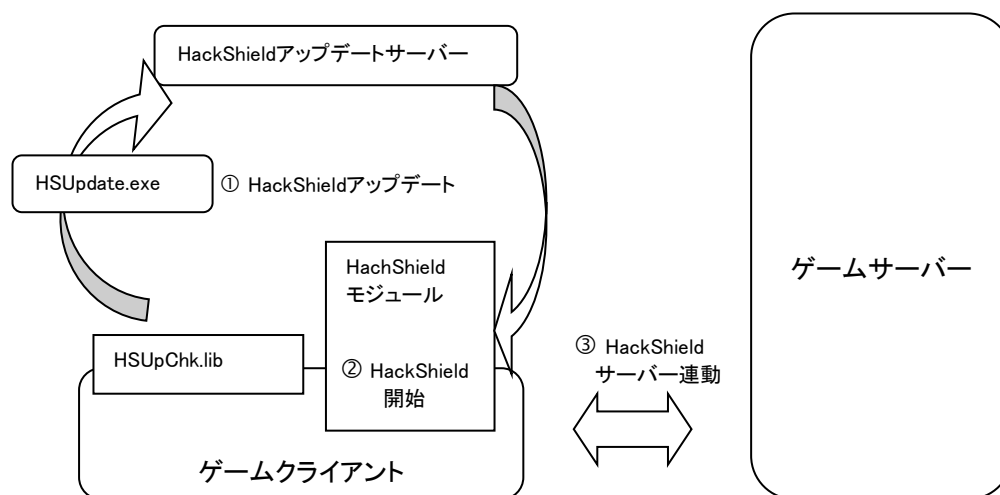


図. 3-1 HackShield アップデート

1. HSUpchk.lib で提供される _AhnHS_HSUpdate API を呼び出すと HackShield アップデートサーバーにアクセスして、HackShield モジュールとエンジンの更新を行います。
2. HShield.lib によって提供される _AhnHS_Initialize 関数と_AhnHS_Start 関数を HackShieldを実行するために呼び出します。
3. ゲームサーバーへのアクセスとのサーバーの相互動作を開始します。

3.2. Application Programming

この章ではどのように HSUpChk.lib よって提供される API を使用してHackShield アップデートを利用するかについて説明します。

参考

本書で使用しているサンプルコードは Microsoft Visual C++ 6.0 を対象とした C/C++ 言語により作成されています。プログラミングに使われている言語は各プログラムの特性やシステム環境によって変更することができます。

プログラミング適用方法

HSUpChk.libを利用してプログラミングを始める前に以下の準備が必要です。

3.2.1.1 アップデート関連ファイル

アップデート関連ファイル

表 3-1 アップデート関連ファイル

ファイル名	インストールフォルダ	説明
HSUpChk.lib	[プログラムソースフォルダ]	サーバーで利用するライブラリ
HSUpChk.h	[プログラムソースフォルダ]	サーバーで利用するヘッダー
AhnUpCtl.dll	[Game]¥HShield	アップデートdll
AhnUpGS.dll	[Game]¥HShield	アップデートdll
HSInst.dll	[Game]¥HShield	UI dll
HSUpdate.env	[Game]¥HShield	アップデート環境設定ファイル
HSUpdate.exe	[Game]¥HShield	アップデート実行ファイル
V3Hunt.dll	[Game]¥HShield	アップデート dll
V3InetGS.dll	[Game]¥HShield	アップデート dll

表 3-2 アップデートサーバーの必須ファイルリスト

パッチセットパス: [HackShield SDK] ¥PatchSet¥

ファイル名	説明
ahn.ui	アップデート情報ファイル(エンジンファイルバージョン情報)
ahn.uic	アップデート情報ファイル(エンジンファイルバージョン情報)
ahni2.dll	アップデートファイル
AhnUpCtl.dll	アップデートファイル
autoup.exe	アップデートファイル
v3bz32.dll	アップデートファイル
patch¥39¥3n.mh-	ヒューリスティックエンジンファイル (圧縮ファイル)
patch¥39¥ahn.ui	アップデート情報ファイル(パッチファイルバージョン情報)
patch¥39¥ahn.uic	アップデート情報ファイル(パッチファイルバージョン情報)
patch¥39¥ahnupctl.dl-	アップデートファイル (圧縮ファイル)
patch¥39¥ahnupgs.dl-	アップデートファイル (圧縮ファイル)
patch¥39¥ehsvc.dl-	HackShieldインターフェースDLLアップデートファイル (圧縮ファイル)
patch¥39¥hshield.da-	HackShield 関連 dat ファイル (圧縮ファイル)
patch¥39¥hsinst.dl-	アップデートファイル (圧縮ファイル)
patch¥39¥hsupdate.ex-	アップデート実行ファイル (圧縮ファイル)
patch¥39¥v3hunt.dl-	アップデートファイル (圧縮ファイル)
patch¥39¥v3inets.dl-	アップデートファイル (圧縮ファイル)
win¥e¥b¥b_echo_sl¥asc_com.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥asc_dh.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥asc_fse.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥asc_intg.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥asc_mmgr.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥asc_unp.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥fse_base.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥fse_fact.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥fse_pe.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥gfs_base.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥gfs_fact.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥gfs_file.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥gfs_mem.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥gfs_os.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥gfs_proc.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_echo_sl¥gfs_util.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)
win¥e¥b¥b_sign_hs¥0asc.sc-	ハッキングツールパターンエンジンファイル (圧縮ファイル)
win¥e¥b¥b_sign_hs¥0secure.sc-	ハッキングツールパターンエンジンファイル (圧縮ファイル)
win¥e¥b¥b_sign_hs¥0sgame.sc-	ハッキングツールパターンエンジンファイル (圧縮ファイル)
win¥e¥b¥b_sign_hs¥0spe3f.sc-	ハッキングツールパターンエンジンファイル (圧縮ファイル)
win¥e¥b¥b_sign_hs¥moduler.sc-	ハッキングツールパターンエンジンファイル (圧縮ファイル)
win¥e¥b¥b_sign_hs¥option.sc-	ハッキングツールパターンエンジンファイル (圧縮ファイル)
win¥e¥b¥b_v3_echo_hs¥v3pro32s.dl-	ハッキングツール検知インターフェースDLL (圧縮ファイル)

表 3-3 アップデートに使われるデータファイル

データファイルパス: [HackShield SDK] ¥ Data¥

ファイル名	説明
noupdate.ui	アップデート機能を一時的にOFFにするために必要なファイル

アップデートが終わった後、以下のファイルが自動で生成されます。

ファイル名	インストールフォルダ	説明
ahn.ui	[HackShield Folder] /Update/	アップデート情報ファイル
ahn.uic	[HackShield Folder] /Update/	アップデート情報ファイル
ahni2.dll	[HackShield Folder] /Update/	アップデート DLL
AhnUpCtl.dll	[HackShield Folder] /Update/	アップデート DLL
autoup.exe	[HackShield Folder] /Update/	アップデート実行ファイル (ファイルパッチ処理)
v3bz32.dll	[HackShield Folder] /Update/	アップデート DLL (ファイル圧縮処理)

3.2.1.2.適用方法

サーバー運営方法

HackShield アップデートサーバーは地域ごとにパブリシャー側の準備・セッティング・運営が必要です。

表 3-4 HackShield アップデートサーバースペック

項目	推奨スペック	最小スペック
Operating system	Windows Server 2003	Windows Server 2000
CPU	Intel Xeon 3.2 Dual	Intel Pentium IV 2.4
RAM	2G 以上	1G 以上
HDD	40G 以上	40G 以上

1. IIS を利用して外部からの接続が可能なHTTP または、FTP で構成してください。

参考

FTPの場合、個別のアカウントを設定または、anonymous接続を許可する必要があります。

また、パッシブモードでのアクセスを可能にする必要があります。

2. [HackShield SDK] ¥PatchSet folder 以下の全てのファイルをサーバーにアップロードしてください。

ファイルのリストは、表 3-2を参照してください。

注意

パッチセットをアップロードするフォルダはサーバーのルートフォルダの下に作成する必要があります。(ルートに直接アップロードしないでください。)

注意

アドレスを設定するときに、以下の点にも注意してください。

アップデートサーバーの URL に“ahnlab”を含めないでください。ワームの一部には“ahnlab”を含むアドレスへのアクセスを妨害するものがあります。

例: ahlalabGame.jp/real/ だとアドレスに“ahnlab”を含むのでアクセスが妨害される。

クライアント適用方法

1. HSUpChk.lib ライブラリファイルを作業するプロジェクトに含む必要があります。
2. HSUpChk.h ファイルを作業するソースファイルに追加してください。

参考

2つのタイプのアップデート API を提供します。

AhnHSUpdate :

基本アップデート機能

AhnHSUpdateEx :
基本アップデート機能 + 環境ファイル改ざん検知 + Hosts ファイルチェック機能

3. アップデートAPI (AhnHS_HSUpdateEx or AhnHS_HSUpdate)を使います。

```
DWORD          dwRet = 0;
TCHAR  szFullFilePath[MAX_PATH]={ 0, };

// HackShieldフォルダのパスを指定します。
_tcscpy ( szFullFilePath, _T( “ .¥¥HShield” ) );

AHNHS_EXT_ERRORINFO HsExtError={ 0, };

// Specify the monitoring information.
HsExtError.szServer = “127.0.0.1”      //monitoring address
HsExtError.szUserId = “Test”           //user ID
HsExtError.szGameVersion = “3.0.0.1” //Game version

// _AhnHS_HSUpdate 関数呼び出し
dwRet = _AhnHS_HSUpdateEx( szFullFilePath, // HackShield フォルダパス
                           1000 * 600,    // アップデート全体のタイムアウト
                           1234,          // Game code
                           AHNHSUPDATE_CHKOPT_HOSTFILE| AHNHSUPDATE_CHKOPT_GAMECODE,
                           HsExtError,
                           1000* 20 ); //サーバー接続タイムアウト
// Ex関数を利用する時には必ずHSupSetEnv.exe設定ツールを使ってenvファイルに
//.ゲームコードを入れてください。
if( dwRet != ERROR_SUCCESS)
{
    // エラー処理
    switch ( dwRet )
    {
        case HSERROR_ENVFILE_NOTREAD:
            ExitClient();
            break;
        case HSERROR_ENVFILE_NOTWRITE:
            ExitClient();
            break;
        case HSERROR_NETWORK_CONNECT_FAIL:
            ExitClient();
            break;
        case HSERROR_HSUPDATE_TIMEOUT:
            ExitClient();
            break;
        case HSERROR_MISMATCH_ENVFILE:
            ExitClient();
    }
}
```

```

        break;
    case HSERROR_HOSTFILE_MODIFICATION:
        ExitClient();
        break;
    ...
}
}

```

```

DWORD          dwRet = 0;
TCHAR  szFullPath[MAX_PATH]={ 0, };

// HackShieldフォルダのパスを指定します。
_tcscpy ( szFullPath, _T( “.%¥HShield” ) );

// _AhnHS_HSUpdate 関数呼び出し
dwRet = _AhnHS_HSUpdate (  szFullPath, // HackShieldフォルダパス
                          1000 * 600,  // アップデート全体のタイムアウト
                          1000* 20 );  // サーバー接続タイムアウト
if( dwRet != ERROR_SUCCESS)
{
    // エラー処理
    switch ( dwRet )
    {
        case HSERROR_ENVFILE_NOTREAD:
            ExitClient();
            break;
        case HSERROR_ENVFILE_NOTWRITE:
            ExitClient();
            break;
        case HSERROR_NETWORK_CONNECT_FAIL:
            ExitClient();
            break;
        case HSERROR_HSUPDATE_TIMEOUT:
            ExitClient();
            break;
        ...
    }
}
}

```

1. [HackShield SDK] ¥Bin¥Win¥x86¥Update フォルダのファイル全部を配布するクライアントの[Game Directory]¥HShield フォルダにコピーしてください。

フォルダ、ファイル構造のコピーが終わるとは、以下ようになります。

サンプル - HackShield アップデートモジュールを利用する時のディレクトリ構造

[Game Directory]¥HShield¥

3n.mhe

AhnRpt.exe

```
ahnrpt.ini
AhnUpCtl.dll
AhnUpGS.dll
BldInfo.ini
EHSvc.dll
hshield.dat
HSLogMgr.exe
HSInst.dll
HSUpdate.env
HSUpdate.exe
V3Hunt.dll
V3InetGS.dll
v3pro32s.dll

[Game Directory]¥HShield¥asc
0asc.scd
0scure.scd
0sgame.scd
0spe3f.scd
asc_com.dll
asc_dh.dll
asc_fse.dll
asc_intg.dll
asc_mmgr.dll
asc_unp.dll
fse_base.dll
fse_fact.dll
fse_pe.dll
gfs_base.dll
gfs_fact.dll
gfs_file.dll
gfs_mem.dll
gfs_os.dll
gfs_proc.dll
gfs_util.dll
moduler.scd
option.scd
```

2. [HackShield SDK]¥Bin¥Win¥x86¥Util¥HSUpSetEnv.exe ファイルを実行します。

注意

HSUpSetEnv.exeは、アップデート構成ファイルを作成するツールです。このツールを配布することはできません。

3. [9.5 HSUpSetEnv ツールの使い方](#) を参照してHSUpSetEnv.exeファイルを作成します。

4. HSUpdate.env ファイルを[GameDirecotry]¥HShield フォルダにコピーします。

注意

アップデートを行うためには_AhnHS_HSUpdate関数の最初のパラメータ([ゲームディレクトリ]¥HShield)のパスに、[HackShield SDK]¥Bin¥Win¥x86¥Update] 内のすべてのアップデートモジュールをコピーする必要があります。

5. スプラッシュイメージを利用する場合は、“splash.jpg” というファイル名で [Game Directory]¥HShield フォルダにイメージを配置してください。

6. アップデート中に表示される画面の右下側のイメージを変更する場合、“hupdate.jpg” というファイル名で [Game Directory]¥HShield フォルダに配置してください。

デフォルトのアップデートイメージの大きさは、200 * 149 ドットです。イメージの下にはプログレスバーが表示されます。



図 3-2 デフォルトHackShieldアップデートイメージ

参考

1. アップデートイメージ

- _ ファイル名: hupdate.jpg (大文字・小文字は無関係)
- _ Size: 200 * 149
- _ 配置先: [Game Directory]¥HShield フォルダ
- _ Description: フォルダにファイルがない場合には、デフォルトの表示になります。

2. スプラッシュイメージ

- _ ファイル名: splash.jpg (大文字・小文字は無関係)
 - _ Size: 290 * 190
 - _ 配置先: [Game Directory]¥HShield フォルダ
 - _ Description: フォルダにファイルがない場合には、デフォルトの表示になります。
-

3.3. Application Programming Interface

AhnHS_HSUpdateEx

DESCRIPTION

HackShield ファイルをアップデートします。HSUpdate.env ファイルと Hosts ファイルをチェックする機能があります。

SYNTAX

```
DWORD _stdcall
_AhnHS_HSUpdateEx(
    LPCTSTR          szUpdateDir
    DWORD            dwTimeOut,
    INT64            i64GameCode,
    AHNHS_EXT_ERRORINFO HsExtErrorInfo,
    DWORD            dwTimeOutPerConnection = 0
);
```

PARAMETERS

Parameter	Value	Description
szUpdateDir	LPCTSTR	更新ファイルを配置したフォルダ
dwTimeOut	DWORD (milliseconds)	アップデート タイムアウト 0 に設定するとタイムアウトが無制限に設定されます。 推奨値は、600,000 (10分)です。特別な理由がない限り、0 をセットしタイムアウトを無制限に設定しないことをお勧めします。
INT64	i64GameCode	HSUpSetEnv.exe ツールを使用して HSUpdate.envファイルに保存したゲームのコードの値を入力します。(モニタリングサーバー使用時は必須。)
DWORD	dwOption	アップデート追加機能オプション AHNHSUPDATE_CHKOPT_HOSTFILE AHNHSUPDATE_CHKOPT_GAMECODE
AHNHS_EXT_ERRORINFO	HsExtErrorInfo	サーバーURL、ユーザーID、ゲームバージョンを含む構造体

dwTimeOutPerConnection	DWORD (milliseconds)	<p>接続中のタイムアウト時間</p> <p>ネットワークの状態に応じて適切な値を設定してください。</p> <p>0 にセットすると接続中のタイムアウトは機能しません。</p>
------------------------	-------------------------	---

szUpdateDir

[in] szUpdateDir 変数には、HackShield 関連ファイルの絶対パスを指定します。このフォルダには、必ずすべてのアップデートファイルが存在する必要があります。GetModuleFileName API を利用して正確な絶対パスを取得することを推奨します。GetCurrentDirectory API では、意図したパスの取得ができない場合があります。

dwTimeOut

[in] 関数を呼び出した後の待機時間です。このパラメーターはのミリ秒単位になっていて、指定した時間を過ぎたら関数呼び出し失敗だと判断します。アップデート失敗時には開発会社のポリシーによってゲームを実行するかを判断してください。
(推奨値は、600,000 (10分)です。)

i64GameCode

[in] アップデートモジュールを利用する時、アップデート環境設置ファイル(HSUpdate.env)の変更を許可しないため、HSUpdate.envファイルに入れたゲームコードと同じゲームコードを入れる必要があります。もし、ゲームコードに差異があるかコードがない場合、HSERROR_MISMATCH_ENVFILE エラーを返します。

ゲームコードはモニタリングサーバーを利用する際に、ゲームを識別するために必須です。

dwOption

[in]_AhnHS_HSUpdateEx 関数の機能を定義するオプションです。

AHNHSUPDATE_CHKOPT_HOSTFILE: Hosts ファイルをチェックしてアップデートサーバー目録の中でHostsファイルに定義されているIP アドレスがあったと判断したら

HSERROR_HOSTFILE_MODIFICATION エラーを返します。

これは、悪意のある人によってアップデートの手順をバイパスするために悪用される可能性がある脆弱性を防ぎます。

AHNHSUPDATE_CHKOPT_GAMECODE: HSUpdate.env ファイルにあるゲームコードと i64GameCode 引数を比較します。比較した結果、同一でない場合は HSERROR_MISMATCH_ENVFILE エラーを返します。

HsExtErrorInfo

[in] サーバー IP、ユーザーアカウント、ゲームバージョンを含む構造体です。

szServer メンバは、HackShield モニタリングサーバの IP アドレスを設定します。

szUserID メンバは、ユーザーアカウント情報を設定します。szGameVersion は、ゲームクライアントのバージョンをセットします。

HackShield アップデートに対してモニタリングサービスの機能を利用する場合、必ずこの構造体に情報を入れる必要があります。もしHackShield アップデートに対してモニタリングサービスの機能を利用しない場合は構造体のメンバを ZeroMemory で初期化してください。

dwTimeOutPerConnection

[in] サーバー接続中の待ち時間の合計です。dwTimeOutPerConnection の時間（ミリ秒）待ってレスポンスがない場合、接続が失敗します。

dwTimeOutPerConnection は HSUpdate.env ファイルに複数のサーバーを設定している場合、すべてのサーバーに同様に反映されます。

0 にセットした場合、タイムアウトは無効になります。

RETURN VALUE

HACKSHIELD_ERROR_SUCESS (Value = 0x00000000)

内容: アップデートに成功しました。

HSERROR_ENVFILE_NOTREAD (Value = 0x30000010)

内容: HSUpdate.env ファイルを読み込めません。

原因: 環境ファイルがないか、HSUpSetEnv.exe ツールのバージョンに互換性がない場合に発生します。

対処方法: 環境ファイルが正しいパスにあるか確認します。

HSUpSetEnv.exe ツールが最新バージョンであるか確認します。

HSERROR_ENVFILE_NOTREADFOUND (Value = 0x30000011)

内容: HSUpdate.env ファイルがありません。

原因: 環境ファイルがないかユーザーにアクセス権がありません。

対処方法: フォルダにファイルがあるか確認します。また、アクセス権限を確認します。

HSERROR_UPDATE_INITIALIZE_FAILED (Value = 0x3000001C)

内容: モジュールの初期化に失敗しました。

原因: HSUpdate.exe を利用してアップデートの初期化中に発生します。

対処方法: このエラーはアップデート処理の内部で発生します。アンラボにお問い合わせ

してください。

HSERROR_ENVFILE_NOTWRITE (Value = 0x30000020)

内容: HSUpdate.env ファイルに書き込めません。

原因: HSUpdate.env ファイルに書き込み権限がないか、アクセスできない場合に発生します。

対処方法: env ファイルのアクセス権限を確認します。同様の問題が発生した場合、一旦ファイルを削除して新たに作成しなおします。

HSERROR_NETWORK_CONNECT_FAIL (Value = 0x30000030)

内容: アップデートサーバーに接続できません。

原因: アップデートサーバーへ接続することができません (ftp/http)。

対処方法: IE や ftp ツールでアップデートサーバーへ接続できるか確認します。

環境ファイルのサーバーアドレス、ID / パスワードを確認します。

一部のワームは、セキュリティプログラムのサイトへのアクセスを制限します。ウイルス対策プログラムでシステムをスキャンして感染の有無を確認します。

サーバーの *.ui ファイルを読み事ができない場合もこのエラーが発生します。

*.ui ファイルを読み込むことが可能か確認します。

HSERROR_LIB_NOTEDIT_REG (Value = 0x30000050)

内容: ネットワークからデータを入力中にエラーが発生しました。

HSERROR_NOTFINDFILE (Value = 0x30000060)

内容: HackShield アップデート関連ファイルが見つかりません。

HSERROR_PROTECT_LISTLOAD_FAIL (Value = 0x30000070)

内容: HSUpdate.pt 認証ファイルが見つかりません。

HSERROR_HSUPDATE_TIMEOUT (Value = 0x30000090)

内容: アップデートタイムアウトが発生しました。

原因: AhnHS_HSUpdate 関数では、タイムアウトを設定できます。アップデートが一定時

間内に終了しないと発生します。また、ネットワークに問題がある場合にも、このエラーが発生します。

HSERROR_STRING_CONVERSION_FAILED (Value = 0x300000B0)

内容: Unicode 文字列の変換でエラーが発生しました。

原因: Unicode 変換する API 関数でエラーが起きました。

対処方法: szUpdateDir の値を確認してください。

HSERROR_MISMATCH_ENVFILE (Value = 0x300000C0)

内容: ゲームクライアントが環境ファイルの記述と違います。

原因: _AhnHS_HSUpdateEx 関数を呼ぶときに指定する i64GameCode と HSUpdate.env ファイルの GameCode が一致しません。

対処方法: HSUpdate.env ファイルと i64GameCode 同一であることを確認してください。この機能は、AHNHSUPDATE_CHKOPT_GAMECODE でオン・オフ切り換えることが可能です。

HSERROR_HOSTFILE_MODIFICATION (Value = 0x300000D0)

内容: Hosts ファイルにアップデート URL が含まれています。

原因: HSUpdate.env にあるアップデートサーバーの IP アドレスが Hosts ファイルに保存されています。

対処方法: HackShield アップデートサーバーの URL が Hosts ファイルにあるか確認します。この機能は、AHNHSUPDATE_CHKOPT_GAMECODE でオン・オフ切り換えることが可能です。

HSERROR_AUTOUPDATE_FAIL (Value = 0x300000E0)

内容: HackShield 自動アップデート の処理中にエラーが発生しました。

原因: このエラーはアップデート処理の内部で発生します。アンラボにお問い合わせください

対処方法:

HSERROR_UPDATE_WIN32_ERROR (Value = 0x3000FFFF)

内容: HackShield 自動アップデート の処理中にエラーが発生しました。

原因: このエラーはアップデート処理の内部で発生します。 アンラボにお問い合わせください。

対処方法:

REMARKS

アップデート関数は、初期化関数（_AhnHS_Initialize）の前に呼び出す必要があります。他のHackShieldの関数と違ってゲームラウンチャのようなプロセスから HackShield の機能とは無関係に実行することが可能です。

_AhnHS_HSUpdate

DESCRIPTION

HackShield ファイルを更新します。

SYNTAX

```
DWORD __stdcall  
_AhnHS_HSUpdate(  
    LPCTSTR szUpdateDir,  
    DWORD dwTimeOut,  
    DWORD dwTimeOutPerConnection = 0  
);
```

PARAMETERS

Parameter	Value	Description
szUpdateDir	LPCTSTR	更新ファイルを配置したフォルダ

dwTimeOut	DWORD (milliseconds)	アップデート タイムアウト 0 に設定するとタイムアウトが無制限に設定されます。 推奨値は、600,000 (10分)です。特別な理由がない限り、0 をセットしタイムアウトを無制限に設定しないことをお勧めします。
dwTimeOutPerConnection	DWORD (milliseconds)	サーバー接続 タイムアウト * ネットワーク環境に合わせて設定 * 0に設定するとサーバーに接続した場合タイムアウトがOFFになります。

szUpdateDir

[in] szUpdateDir 変数には、HackShield 関連ファイルの絶対パスを指定します。このフォルダには、必ずすべてのアップデートファイルが存在する必要があります。GetModuleFileName API を利用して正確な絶対パスを取得することを推奨します。GetCurrentDirectory API では、意図したパスの取得ができない場合があります。

dwTimeOut

[in] 関数を呼び出した後の待機時間です。このパラメーターはのミリ秒単位になっていて、指定した時間を過ぎたら関数呼び出し失敗だと判断します。アップデート失敗時には開発会社のポリシーによってゲームを実行するかを判断してください。
(推奨値は、600,000 (10分)です。)

dwTimeOutPerConnection

[in] サーバー接続時の待ち時間です。このパラメーターはのミリ秒単位になっていて、指定した時間を過ぎたらサーバー接続失敗だと判断します。HSUpdateEnv.exeを使ってHSUpdate.env ファイルに複数のサーバーを設定している場合、すべてのサーバーに対してdwTimeOutPerConnectionの値が同様に反映されます。

0 にセットした場合、タイムアウトは無効になります。

RETURN VALUE

HACKSHIELD_ERROR_SUCESS (Value = 0x00000000)

内容: アップデートに成功しました。

HSERROR_ENVFILE_NOTREAD (Value = 0x30000010)

内容: HSUpdate.env ファイルを読み込めません。

原因: env環境ファイルがないか、HSUpSetEnv.exe ツールのバージョンに互換性がない場合に発生します。

対処方法: 環境ファイルが正しいパスにあるか確認します。
HSUpSetEnv.exe ツールのバージョンが最新であるか確認します。

HSERROR_ENVFILE_NOTWRITE (Value = 0x30000020)

内容: HSUpdate.env ファイルに書き込めません。

原因: env ファイルに書き込み権限がないか、アクセスできない場合に発生します。

対処方法: HSUpdate.env ファイルのアクセス権限を確認します。同様の問題が発生した場合、一旦ファイルを削除して新たに作成しなおします。

HSERROR_NETWORK_CONNECT_FAIL (Value = 0x30000030)

内容: サーバーに接続できません。

原因: アップデートサーバーへ接続することができません(ftp/http)。

対処方法:

IE や ftp ツールでアップデートサーバーへ接続できるか確認します。

Env環境ファイルのサーバーアドレス、ID / パスワードを確認します。

一部のワームは、セキュリティプログラムのサイトへのアクセスを制限します。ウイルス

対策プログラムでシステムをスキャンして感染の有無を確認します。

*.ui ファイルを読み込むことが可能か確認します。

HSERROR_LIB_NOTEDIT_REG (Value = 0x30000050)

内容: ネットワークからデータを入力中にエラーが発生しました。

HSERROR_NOTFINDFILE (Value = 0x30000060)

内容: HackShield アップデート関連ファイルが見つかりません。

HSERROR_PROTECT_LISTLOAD_FAIL (Value = 0x30000070)

内容: HSUpdate.pt 認証ファイルが見つかりません。

HSERROR_HSUPDATE_TIMEOUT (Value = 0x30000090)

内容: アップデートタイムアウトが発生しました。

原因: _AhnHS_HSUpdate 関数では、タイムアウトを設定できます。アップデートが一定時間内に終了しないと発生します。また、ネットワークに問題がある場合にも、このエラーが発生します。

HSERROR_AUTOUPDATE_FAIL (Value = 0x300000E0)

内容: HackShield 自動アップデート の処理中にエラーが発生しました。

原因: このエラーはアップデート処理の内部で発生します。アンラボにお問い合わせください。

REMARKS

アップデート関数は、初期化関数 (_AhnHS_Initialize) の前に呼び出す必要があります。他のHackShieldの関数と違ってゲームラウンチャのようなプロセスから HackShield の機能とは無関係に実行することが可能です。

4. 拡張サーバー連動機能

4.1. 概要

拡張サーバー連動（AntiCpX）はファイル/メモリ操作検知機能を持つ旧・サーバー連動 SDK（Software Development Kit）から一段階発展した SDK です。大きく向上した点は旧・サーバー連動では関数単位でメモリ上にロードされたコードを保護していたのに対し、拡張サーバー連動ではコード領域全体を保護している点です。

機能

拡張サーバー連動（AntiCpX）は旧サーバー連動機能を備えており、メモリ完全性機能が強化されています。

ゲームクライアントファイル完全性チェック

サーバーでは AntiCpXSvr.dll(libanticpxsvr.so, libanticpxsvr_st.a) と AntiCpXSvr.h を利用し、クライアントでは HackShield モジュールである HShield.lib と HShield.h を使ってサーバーとクライアント間で通信することにより、リアルタイムでゲーム実行ファイルが操作されていないかを確認することができます。

注意

ゲームクライアントがウイルスに感染された場合にも拡張サーバー連動で検知することができます。これをゲーム会社のポリシーに合わせてコールバックに対する終了処理をする必要があります。

パケット完全性機能

パケットをキャプチャし、特定の状況に合わせてパケットを生成して正常に動作するハッキングを防ぐため、パケットの完全性を保障するモジュールが内部的に動作します。ネットワークでパケットをキャプチャして生成するなどの方法を遮断することができます。なお、この機能はサーバー連動クラック防止と関連したメッセージに限り保護します。

HackShieldの正常動作確認

サーバーでは AntiCpXSvr.dll (libanticpxsvr.so, libanticpxsvr_st.a)と AntiCpXSvr.h を利用し、クライアントでは HackShield モジュールである HShield.lib と HShield.h を使ってサーバーとクライアント間で通信することにより、簡単に HackShield の動作状態を確認

することができます。サーバーは HackShield が動作してからクエリメッセージを転送する必要があります。

メモリー完全性機能（拡張サーバー連動強化機能）

サーバーでは AntiCpXSvr.dll (libanticpxsvr.so, libanticpxsvr_st.a)と AntiCpXSvr.h を利用し、クライアントでは HackShield モジュールである HShield.lib と HShield.h を使ってサーバーとクライアント間で通信することにより、リアルタイムでゲームプロセスのメモリーが操作されていないかを検証することができます。拡張サーバー連動では機能が強化され、旧・サーバー連動では関数単位でメモリー上のコードを保護していたのに対し、拡張サーバー連動ではコード領域全体を保護します。

HackShieldエンジン完全性確認

サーバーでは AntiCpXSvr.dll (libanticpxsvr.so, libanticpxsvr_st.a) と AntiCpXSvr.h を利用し、クライアントでは HackShield モジュールである HShield.lib と HShield.h を使ってサーバーとクライアント間で通信することにより、ヒューリスティックエンジン(3N.mhe) ファイルの完全性を検証することができます。

特徴

インターフェイス関数(API)の提供

拡張サーバー連動 (AntiCpX) の機能を使ってその実行結果の値を確認できるようインターフェース DLL とライブラリを提供します。開発者は提供されているインターフェース DLL とライブラリを使い、固有ポリシーに従ってゲームファイルの完全性や HackShield が正常に動作しているかを確認することができます。

HSBデータファイル情報生成プログラム提供

拡張サーバー連動 (AntiCpX) の HSBGen.exe はクライアントとサーバーが通信してクライアントプログラムのファイル/メモリーの完全性の有無を判断する基準となる各種ファイル情報やメモリー情報を生成して保存します。ゲーム開発会社の特性に合わせてサーバーでファイルやメモリー情報を保存することができ、このデータを使ってサーバー連動のファイル/メモリークラック防止機能が動作するようになります。詳しい使用方法是「9.3 HSBGen ツール (HackShield 専用、4.2 以降のバージョン)」で説明します。

拡張サーバー連動関連ファイル

¥Bin¥Win¥[Platform]¥AntiCrack¥AntiCpXSvr.dll : 拡張サーバー連動 dll ファイル
¥Bin¥Win¥x86¥AntiCrack¥HSBGen.exe : .hsb ファイル作成ツール
¥Bin¥Win¥x86¥AntiCrack¥HSPub.key : サーバー連動認証キーファイル
¥Bin¥Win¥x86¥HShield¥3n.mhe : ヒューリスティックエンジン バージョン管理ファイル
¥Bin¥Win¥x86¥HShield¥hshield.dat : HackShield バージョン管理ファイル
¥Include¥AntiCpXSvr.h : 拡張サーバー連動ヘッダーファイル

¥Lib¥Win¥x86¥AntiCpXSvr.lib :
 拡張サーバー連動ライブラリファイル (for Windows 32Bit)

¥Lib¥Win¥x86¥AntiCpXSvr.lib :
 拡張サーバー連動ライブラリファイル (for Windows 64Bit)

¥Bin¥Linux¥[Platform]¥AntiCrack¥libanticpxsvr.so :
 拡張サーバー連動ダイナミックライブラリファイル (for Linux)

¥Bin¥Solaris¥x86¥AntiCrack¥libanticpxsvr.so:
 拡張サーバー連動ダイナミックライブラリファイル (for Solaris)

¥Lib¥Linux¥[Platform] ¥libanticpxsvr_st.a :
 拡張サーバー連動スタティックライブラリファイル (for Linux)

¥Lib¥Solaris¥x86¥AntiCrack¥libanticpxsvr_st.a :
 拡張サーバー連動スタティックライブラリファイル (for Solaris)

※Solaris用ライブラリは、32ビット x86プラットフォームのみサポートします。

システムアーキテクチャ

拡張サーバー連動 (AntiCpX) は AntiCpXSvr.dll (libanticpxsvr.so, libanticpxsvr_st.a)、HShield.lib、EHSvc.dll を提供し、サーバーとクライアントに DLL とライブラリ形態で適用します。拡張サーバー連動 (AntiCpX) の全体構造および動作原理は次のとおりです。
 :

AntiCpXSvr.dll (インターフェース dll)

Windows.用ダイナミックライブラリファイル

サーバーで使用します。サーバーで要求メッセージを生成し、クライアントから受け取ったメッセージを使ってクライアントのファイルやメモリが正常に動作しているかを確認する API を提供します。

AntiCpXSvr.lib

Windows.用スタティックライブラリファイル

サーバーで使用します。サーバーで要求メッセージを生成し、クライアントから受け取ったメッセージを使ってクライアントのファイルやメモリが正常に動作しているかを確認する API を提供します。

libanticpxsvr.so

Linux用及びSolaris.用ダイナミックライブラリファイル

サーバーで使用します。サーバーで要求メッセージを生成し、クライアントから受け取ったメッセージを使ってクライアントのファイルやメモリが正常に動作しているかを確認する API を提供します。

libanticpxsvr_st.a

Linux用及びSolaris.用スタティックライブラリファイル

サーバーで使用します。サーバーで要求メッセージを生成し、クライアントから受け取ったメッセージを使ってクライアントのファイルやメモリが正常に動作しているかを確認するAPIを提供します。

HShield.lib (HackShield ライブラリ)

クライアントで使用します。サーバーから転送された要求メッセージを受けて要求された内容を実行し、結果として得た各種情報を暗号化してサーバーに送るデータを生成するAPIを提供します。

HSBGen.exe (ファイル情報生成プログラム)

サーバーで使用するファイル情報やメモリ情報のファイルを生成してゲームファイルの完全性の有無や実行するゲームのメモリの完全性の有無を判断できるようにします。ファイル情報データは生成する度に異なるため、ゲームサーバーが複数であればサーバーごとに別々適用することで高いセキュリティを保つことができます。

AntiCpx.hsb (クライアント CRC ファイル)

HSBGen.exe.で作成されます。クライアントプログラムの整合性を検証するために使用します。

HSPub.key (サーバー連動認証キーファイル)

アクセスしてきたクライアントのHackShieldを認証するために使用します。AntiCpx.hsb ファイルと同一フォルダに配置する必要があります。

3n.mhe (ヒューリスティックエンジンファイル)

クライアントで使用されるエンジンファイルです。サーバー連動機能利用時にAntiCpX.hsb ファイルと同じフォルダに配置されるとクライアントは、旧バージョンの3n.mhe ファイルとの接続を切断します。

hshield.dat (HackShieldバージョンファイル)

クライアントで使用されるデータファイルです。サーバー連動機能利用時にAntiCpX.hsb ファイルと同じフォルダに配置されるとクライアントでは、旧バージョンの hshield.dat.hsb ファイルとの接続を切断します。

参考

3n.mhe または hshield.dat ファイルが hsb ファイルと同一のサーバーフォルダにある場合、旧バージョンのHackShieldを使うクライアントの接続を切断することができます。(3n.mhe または hshield.dat をアップロードした時にサーバーをリスタートする必要はありません。)

下の表のように“Disconnected”の場合、
_AhnHS_VerifyResponseEx 関数からANTICPX_RECOMMAND_CLOSE_SESSION
(細部エラー: ERROR_ANTICPXSVR_INVALID_ENGINE_VERSION
/ERROR_ANTICPXSVR_INVALID_HACKSHIELD_VERSION) エラーを返すとクライアント
コネクションを切断してください。

表 4-1 サーバー連動機能バージョン管理

ファイル名	バージョン比較	接続状態
3n.mhe	Client < Server	Disconnected
	Client ≥ Server	Connected
hshield.dat	Client < Server	Disconnected
	Client ≥ Server	Connected

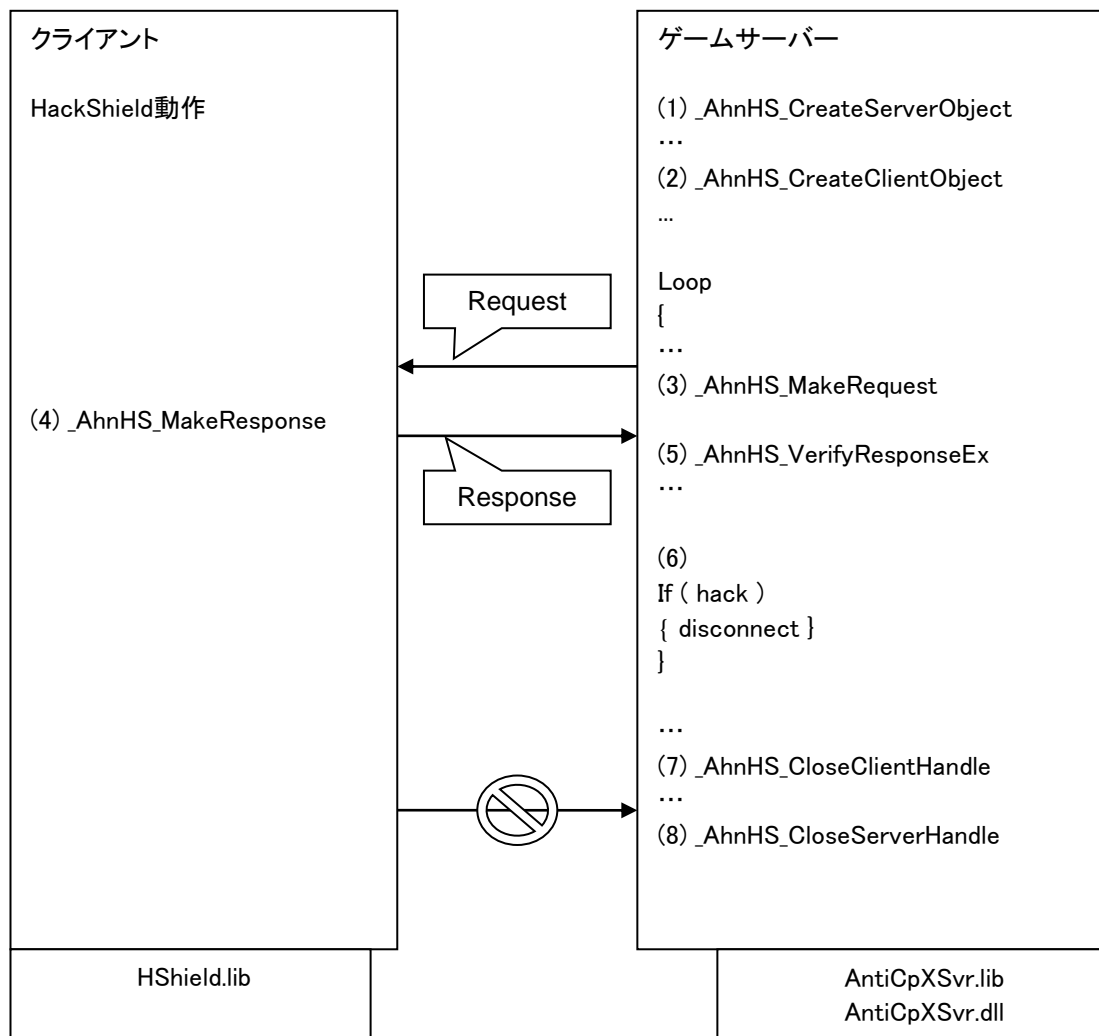


図. 4-1 AntiCpXの動作

ゲームサーバー起動時に `_AhnHS_CreateServerObject` 関数から `AHNHS_SERVER_HANDLE` を生成します。以降、各クライアントがアクセスする度に `_AhnHS_CreateClientObject` 関数とサーバーハンドルをパラメータとして与えて `AHNHS_CLIENT_HANDLE` を生成します。クライアントの偽造/改ざんの有無を監視するため `_AhnHS_MakeRequest` 関数を使って要求メッセージを定期的に生成し、転送します。

クライアントはサーバーの要求メッセージに適切な応答メッセージを生成します。応答メッセージは `_AhnHS_MakeResponse` 関数から生成します。クライアントの応答メッセージが有効であるか `_AhnHS_VerifyResponseEx` 関数で確認します。

注意

`_AhnHS_MakeResponse`関数から生成された応答メッセージがサーバー側に送信され、`_AhnHS_VerifyResponseEx` 関数が呼び出されるまでの間、`_AhnHS_MakeRequest` 関数が呼び出されていないようにする必要があります。

例外(Exception)が発生した場合、ダンプファイルを作成します。

拡張サーバー API 実行中に例外が発生した場合、レジストリを設定することでダンプファイルを作成することができます。レジストリの値はデフォルトではダンプファイルを作成しない設定になっています。

(注意) Windows OSのみ

HKLM\SOFTWARE\AhnLab\HShield\Dump キー関連の設定値

- DumpOnOff
- DumpType
- DumpPath

HKLM\SOFTWARE\AhnLab\HShield\Dumpキーは例外が発生した時に生成されるキーですが、DumpOnOff・DumpType・DumpPathの値は開発者が直接生成して入力する必要があります。

DumpOnOff (DWORD 型)

- 0x0 : ダンプを作成しません。
- 0x1 : ダンプを作成します。

DumpType (DWORD 型)

MINIDUMP_TYPE構造体に定義された値を参考して設定してください。
(MSDN Library参照)

- 0x2 : MiniDumpWithFullMemory (問題を解析するために、この設定を推奨します。)
- 0x1 : MiniDumpNormal

DumpPath (文字列型)

ダンプを保存するフォルダを定義します。フォルダが存在しない場合、アクセスできない場合は、現在ゲームサーバーが実行されたパスにダンプファイルが保存されます。

4.2. Application Programming

拡張サーバー連動（AntiCpX）の API を使ってファイルやメモリの完全性確認を実行する方法について説明します。

参考

本書で使用しているサンプルコードは Microsoft Visual C++ 6.0 を対象とした C/C++ 言語により作成されています。プログラミングに使われている言語は各プログラムの特性やシステム環境によって変更することができます。

プログラミング適用方法

拡張サーバー連動（AntiCpX）を利用してプログラミングする前に、以下のことを確認します。:

1. AntiCpXSvr関連ファイル

AntiCpXSvr関連ファイル

表 4-2 AntiCpXSvr関連ファイル

ファイル名	インストールフォルダ	説明
AntiCpXSvr.h	[プログラムソースフォルダ]	サーバー用ヘッダーファイル
AntiCpXSvr.dll	[プログラムソースフォルダ]	サーバー用DLL ファイル
AntiCpXSvr.lib	[プログラムソースフォルダ]	AntiCPXSvr.dllインポートライブラリ (Windows用)
Libanticpxsvr.so	[プログラム実行フォルダ]	Linux/Solaris 拡張サーバーモジュール (Dynamic ライブラリファイル)
Libanticpxsvr_st.a	[プログラム実行フォルダ]	Linux/Solaris 拡張サーバーモジュール (Static ライブラリファイル)
AntiCpx.hsb	[プログラム実行フォルダ]	クライアント整合性検証ファイル
HSPub.key	[プログラム実行フォルダ]	サーバー連動認証キーファイル
3n.mhe	[HSBファイルがあるフォルダ]	ヒューリスティックエンジンバージョン管理ファイル
hshield.dat	[HSBファイルがあるフォルダ]	HackShield バージョン管理ファイル
HShield.h	[プログラムソースフォルダ]	クライアント用ヘッダーファイル
HShield.lib	[プログラムソースフォルダ]	クライアント用ライブラリファイル
EhSvc.dll	[プログラムソースフォルダ]	クライアント用DLL ファイル

4.2.1.3.適用方法

サーバー適用方法

1. HSBGen.exe を使用して AntiCpx.hsb を作成します。
([10.5.HSBGen Tool](#)参照)
2. HSPub.key ファイルを AntiCpx.hsb ファイルがあるフォルダにコピーします。
3. 必須ではありませんが、HackShield モジュールのバージョン管理の観点から、3n.mhe ファイルとhshield.dat ファイルをhsb ファイルと同一フォルダにコピーすることをお勧めします。

参考

3n.mhe and hs3n.mhe and hsサーバーのhsbが存在するパスに予めクライアントの3N.mheファイルとHShield.datファイルをアップロードしてクライアントのファイルと比較します。それでバージョンが違った場合、クライアントの接続を切断することができます。

4. ゲームサーバー起動時に _AhnHS_CreateServerObject 関数から AHNHS_SERVER_HANDLE を生成します。このハンドルはゲームサーバーが終了するまで維持する必要があります。
(参考: サーバーハンドルは、次の2段階のクライアントハンドルを生成するのに 使用します)
5. 各クライアントが接続する度に _AhnHS_CreateClientObject 関数とサーバーハンドルをパラメータとして与え、AHNHS_CLIENT_HANDLE を生成します。このハンドルはクライアントのネットワークセッションの間、維持されます。
(参考: クライアントハンドルは次の3段階の要求メッセージを生成するのに使用します。)
6. クライアントの偽造/改ざんの有無を監視するために要求メッセージを生成して転送します。要求メッセージは _AhnHS_MakeRequest 関数で生成し、クライアントハンドルをパラメータとして使用します。
7. クライアントはサーバーの要求メッセージに適切な応答メッセージを生成します。応答メッセージは _AhnHS_MakeResponse 関数で生成します。
8. クライアントの応答メッセージが有効であるか確認します。確認は _AhnHS_VerifyResponse 関数で行います。
9. _AhnHS_VerifyResponse 関数で ANTICPX_RECOMMAND_CLOSE_SESSION 値のエラーコードが発生した場合にはエラーコードに従って適切なエラー処理を行った後、ゲームクライアントの接続を中断します。

注意

(AhnHS_VerifyResponseEx または _AhnHS_VerifyResponse を適用してください。)

10. クライアントの接続を切る時(セッションが終了する時)には第 2 段階で生成したクライアントハンドルを閉じます。
11. サーバープロセスが終了する時には、第1段階で生成したサーバーハンドルを閉じます。

(この時クライアントハンドルがすべて閉じた状態である必要があります)

クライアント適用方法

1. HShield.lib ファイルを作業するプロジェクトに追加します。
2. 提供されている HShield.h ファイルを作業するソースに追加します。。
3. サーバーから暗号化されたバージョン情報要求メッセージを受け取ります。
4. AhnHS_MakeResponse を呼び出してサーバーに送る暗号化されたバージョン情報応答メッセージを生成します。
5. 暗号化された応答メッセージをサーバーに転送します。

サーバー連想サイクル

説明: 1サイクルは以下のような流れで構成されます。:

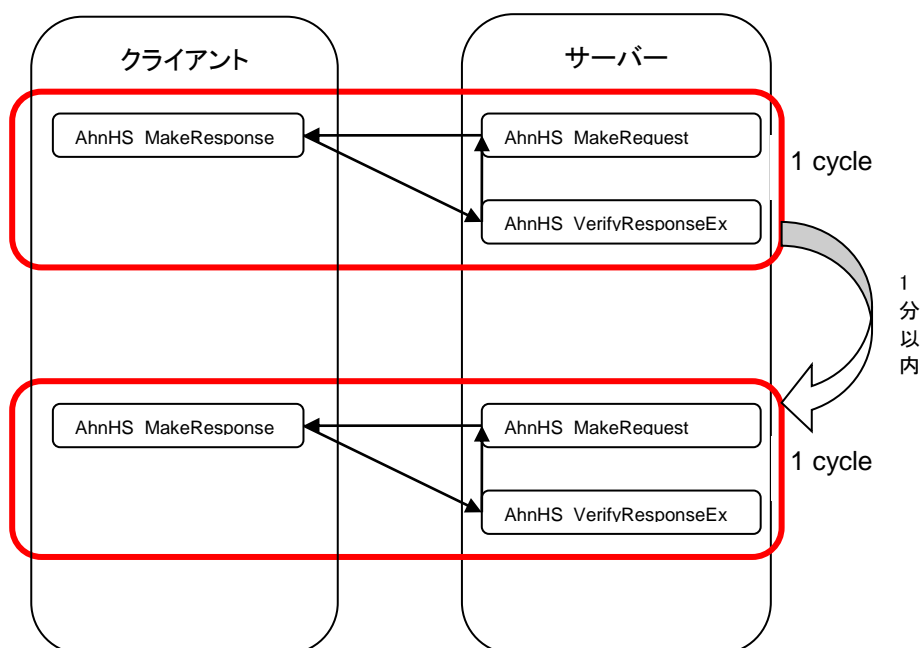
サーバーがリクエストメッセージを送信します。

クライアントは応答メッセージを作成し、サーバーへ送信します。

その結果サーバーは、レスポンスメッセージを受信します。

サイクル: 1サイクルが 1 分以内に終了することを推奨します。

(ゲームの特性によりもっと長いサイクルタイムが必要な場合があります。しかし、その分ハッキングに対する危険性も増します。)



4.3. Application Programming Interface

AhnHS_CreateServerObject

DESCRIPTION

HSBGen.exe で生成した .hsb ファイルをロードしてサーバーハンドル (Server Handle) を生成します。普通は 1 つのゲームをサービス提供するサーバープロセスでサーバーハンドルを 1 つ生成してゲームサーバープロセスが終了するまで維持します。

SYNTAX

```
AHNHS_SERVER_HANDLE _stdcall  
_AhnHS_CreateServerObject (  
    IN const char *pszFilePath  
);
```

PARAMETERS

Parameter	Value	Description
pszFilePath	const char *	HackShield Briefcase (.hsb) ファイルのフルパス

RETURN VALUE

サーバーハンドルが正常に作成できなかった場合 ANTICPX_INVALID_HANDLE_VALUE が返ります。可能性として以下のようなケースが考えられます。:

- ① HackShield Briefcase (.hsb) ファイルが正しくないか存在しない場合。
- ② HSPub.key ファイルがパスに存在しない。
- ③ システムリソース (メモリ) が不足している。
- ④ HSBファイルが存在しないか、アクセス権限がない場合。
- ⑤ _AhnHS_CreateServerObject 関数を実行中に例外が発生した場合。
(例外が発生すると、MiniDump ファイルを生成します。〈※Windows OS の場合のみ〉
‘CREATESERVEROBJECT_year_month_date hour-minute-second.dmp’ という
ファイル名でダンプファイルが保存されます。
そのダンプファイルをアンラボに送信してください。)

Example

_AhnHS_CreateServerObject 関数を呼び出したサンプルは次のとおりです。

サンプル

```
// ファイルのパスは ‘/’ ではなくて ‘¥’ で記録する必要があります。  
//ファイル名までパスを記録してください。
```

```
strcpy(g_szHsbFilePath,  
      “C:¥¥GameServer¥¥HShield¥¥anticpx.hsb” );  
  
hServer = _AhnHS_CreateServerObject (g_szHsbFilePath);  
  
if ( hServer == ANTICPX_INVALID_HANDLE_VALUE )  
{  
    // エラー処理  
}
```

_AhnHS_CloseServerHandle

DESCRIPTION

サーバーハンドル (Server Handle) を閉じます。

SYNTAX

```
void __stdcall  
_AhnHS_CloseServerHandle (  
    IN AHNHS_SERVER_HANDLE hServer  
);
```

PARAMETERS

Parameter	Value	Description
pszFilePath	AHNHS_SERVER_HANDLE	_AhnHS_CreateServerObject 関数で生成したハンドル

RETURN VALUE

None.

Example

_AhnHS_CloseServerHandle 関数を呼び出したサンプルは次のとおりです。

サンプル

```
_AhnHS_CloseServerHandle ( hServer );
```

AhnHS_CreateClientObject

DESCRIPTION

サーバーハンドルの入力を受けて、クライアントハンドルを生成します。クライアントが接続する度に生成し、セッションが続いている間ハンドルを閉じずに再度使用します。

SYNTAX

```
AHNHS_CLIENT_HANDLE  
_AhnHS_CreateClientObject (  
    IN AHNHS_SERVER_HANDLE hServer  
);
```

PARAMETERS

Parameter	Type	Description
hServer	AHNHS_SERVER_HANDLE	サーバーハンドル

RETURN VALUE

クライアントハンドル
サーバーハンドルが正常に作成されていない場合、
ANTICPX_INVALID_HANDLE_VALUE が返ってきます。可能性として以下のようなケースが考えられます。:

- ① hServer 引数が不正な場合、
_AhnHS_CreateServerObject 関数によってhServerサーバーハンドルが正常に作成されたか確認してください。
- ② _AhnHS_CreateServerObject 関数を実行中に例外が発生した場合。
(例外が発生すると、MiniDump ファイルを生成します。〈※Windows OS の場合のみ〉
‘CREATESEVEROBJECT_year_month_date hour-minute-second.dmp’ という
ファイル名でダンプファイルが保存されます。
そのダンプファイルをアンラボに送信してください。)

Example

_AhnHS_CreateClientObject 関数を呼び出したサンプルは次のとおりです。

サンプル

```
hClient = _AhnHS_CreateClientObject ( hServer );  
  
if ( hClient == ANTICPX_INVALID_HANDLE_VALUE )  
{  
    // エラー処理  
}
```

_AhnHS_CloseClientHandle

DESCRIPTION

生成したクライアントハンドルはクライアントのセッションが終了する時に開放する必要があります。この時、クライアントハンドルの生成に使われたメモリやシステムサポートを開放することになります。

SYNTAX

```
void __stdcall  
_AhnHS_CloseClientHandle (  
    IN AHNHS_CLIENT_HANDLE hClient  
);
```

PARAMETERS

Parameter	Type	Description
hClient	AHNHS_CLIENT_HANDLE	クライアントハンドル

RETURN VALUE

None.

Example

_AhnHS_CloseClientHandle 関数を呼び出したサンプルは次のとおりです。

サンプル

```
_AhnHS_CloseClientHandle ( hClient );
```

AhnHS_MakeRequest

DESCRIPTION

現在のセッションに合ったクライアントハンドルを入力して要求メッセージを生成します。要求メッセージは `AHNHS_TRANS_BUFFER` 構造体で出力されます。メンバ変数値は次のとおりです。:

```
typedef struct _AHNHS_TRANS_BUFFER
{
    unsigned short nLength;
    unsigned char byBuffer[ANTICPX_TRANS_BUFFER_MAX]; // 送受信パケットの
    最大バッファサイズ
} AHNHS_TRANS_BUFFER, *PAHNHS_TRANS_BUFFER;
```

`nLength` ; 要求メッセージ生成に使用したバッファの長さ

`byBuffer` ; 要求メッセージ生成に使用できる最大バイトバッファ

注意

`byBuffer`は要求メッセージ生成に使用することが可能な最大バッファサイズですので、ネットワークに送信する時、`nLength`まで送信する必要があります。

SYNTAX

```
unsigned long _stdcall
_AhnHS_MakeRequest (
    IN AHNHS_CLIENT_HANDLE hClient,
    OUT PAHNHS_TRANS_BUFFER pRequestBuffer
);
```

PARAMETERS

Parameter	Type	Description
<code>hClient</code>	<code>AHNHS_CLIENT_HANDLE</code>	クライアントハンドル
<code>pRequestBuffer</code>	<code>PAHNHS_TRANS_BUFFER</code>	転送データのバッファ/サイズ

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

ERROR_ANTICPXSVR_INVALID_PARAMETER (Value = 0xE9040003)

内容: 入力に誤りがあります。

原因: hClient と pbyResponse が NULL です。

対処方法: hClient と pbyResponse が NULL でないか確認します。

ERROR_ANTICPXSVR_BAD_FORMAT (Value = 0xE9040004)

内容: HSB ファイルの読み込みに失敗しました。

原因: HSB ファイルが正しく作成されていません。

対処方法: 最新バージョンの HSBGen.exe を使用して HSBファイルを再度作成してください。

ERROR_ANTICPXSVR_NOT_YET_RECEIVED_RESPONSE (Value = 0xE9040005)

内容: 要求メッセージに対する応答を受け取っていません。

原因: _AhnHS_MakeRequest 関数を呼び出すと以下の手順が実行されます。:

- 1) クライアントから Ack メッセージを受信します。
- 2) _AhnHS_VerifyResponse 関数を呼び出します。
- 3) _AhnHS_MakeRequest 関数を呼び出します。同期が成功しない場合は、トラブルが発生している可能性があります。

対処方法: クライアントセッションの同期に問題がないか確認して下さい。

ERROR_ANTICPXSVR_NOT_ENOUGH_MEMORY (Value = 0xE9040007)

内容: メモリが不足しています。

原因: サーバーのメモリが不足しています。

対処方法: サーバー上でメモリリークが発生していない確認します。

ERROR_ANTICPXSVR_BAD_MESSAGE (Value = 0xE9040008)

内容: バッファの暗号化に失敗しました。

原因: 送信バッファの暗号化に失敗しました。

対処方法: 上記エラーが発生するときは、システムアーキテクチャに問題がある場合があります。アンラボにお問い合わせください。

ERROR_ANTICPXSVR_MAKEREQ_EXCEPTION (Value = 0xE9040014)

内容: _AhnHS_MakeRequest 関数の実行中に例外が発生しました。
(このエラーは Windowsバージョンのモジュールのみで発生します。)

原因: 例外の原因には、複数の問題が考えられます。

対処方法: ミニダンプファイルを 'MAKEREQUEST_year_month_date hour-minute-second.dmp' というファイル名で生成します。生成したダンプファイルをアンラボに送ってください。

Example

_AhnHS_MakeRequest関数を呼び出したサンプルは次のとおりです。

サンプル

```
AHNHS_TRANS_BUFFER stReqTransBuf;

ulRet = _AhnHS_MakeRequest ( hClient, &stReqTransBuf );

if ( ulRet != ERROR_SUCCESS )
    return ulRet;

bytesSent = send( ConnectSocket, stReqTransBuf.byBuffer, stReqTransBuf.nLength, 0 );

...
```

注意

ゲームサーバーにユーザーが正常に接続した後ですぐ_AhnHS_MakeRequest 関数を呼び出してハッキング検視ができるようすることをお勧めします。

AhnHS_VerifyResponseEx

DESCRIPTION

_AhnHS_MakeRequest 関数からの要求（メッセージ）に対するクライアントの応答が正しいかを確認する関数です。
（_AhnHS_VerifyResponse() 関数は内部で呼ばれます。）

SYNTAX

```
unsigned long _stdcall  
_AhnHS_VerifyResponseEx (  
    IN AHNHS_CLIENT_HANDLE hClient,  
    IN unsigned char *pbyResponse,  
    IN unsigned long nResponseLength,  
    OUT unsigned long *pnErrorCode  
);
```

PARAMETERS

Parameter	Type	Description
hClient	AHNHS_CLIENT_HANDLE	クライアントハンドル
pbyResponse	char *	クライアントから受け取るデータのバッファー
nResponseLength	unsigned long	クライアントから受け取るデータの長さ
pnErrorCode	Unsigned long	_AhnHS_VerifyResponse() 関数の戻り値

RETURN VALUE

ANTICPX_RECOMMAND_CLOSE_SESSION (Value = 101)

内容: クライアントでハッキングが検出されました。ゲームサーバーとの接続を遮断する必要があります。

（クライアントからこの値が返ってくるハッキングの多くは、_AhnHS_VerifyResponsef() のエラー値の中でクライアントとの接続を終了する必要があったと案内したエラーです。）

内部で_AhnHS_VerifyResponse関数を呼び出して該当関数の戻り値が下記の値リストの中にある場合、ANTICPX_RECOMMAND_KEEP_SESSIONをリターンします。

- ERROR_ANTICPXSVR_BAD_MESSAGE
- ERROR_ANTICPXSVR_REPLY_ATTACK

- ERROR_ANTICPXSVR_UNKNOWN_CLIENT
- ERROR_ANTICPXSVR_HSHIELD_FILE_ATTACK
- ERROR_ANTICPXSVR_CLIENT_FILE_ATTACK
- ERROR_ANTICPXSVR_MEMORY_ATTACK
- ERROR_ANTICPXSVR_OLD_VERSION_CLIENT_EXPIRED
- ERROR_ANTICPXSVR_NANOENGINE_FILE_ATTACK
- ERROR_ANTICPXSVR_INVALID_HACKSHIELD_VERSION
- ERROR_ANTICPXSVR_INVALID_ENGINE_VERSION
- ERROR_ANTICPXSVR_VERIFY_EXCEPTION
- ERROR_ANTICPXSVR_INVALID_PARAMETER
- ERROR_ANTICPXSVR_ABNORMAL_HACKSHIELD_STATUS
- ERROR_ANTICPXSVR_DETECT_CALLBACK_IS_NOTIFIED

原因: クライアントでハッキングが検出されました。

ANTICPX_RECOMMAND_KEEP_SESSION (Value = 102)

内容: クライアントからの応答内容に問題はありません。通常はゲームサーバーとの接続を維持して問題ありません。

Example

_AhnHS_VerifyResponseSet 関数を呼び出したサンプルは次のとおりです。

サンプル

```

DWORD dwError = 0;
ulRet = _AhnHS_VerifyResponseEx ( hClient,
stResponseBuf.byBuffer,
stResponseBuf.nLength,
&dwError );

if ( ulRet == ANTICPX_RECOMMAND_CLOSE_SESSION )
{
    Log ("[AVREx] Disconnect the Client: 0x%x", dwError );
    // Client Out -> Call _AhnHS_CloseClientHandle を
    // 呼び出して hClient ハンドルを閉める
}

```

AhnHS_VerifyResponse

DESCRIPTION

_AhnHS_MakeRequest 関数からの要求（メッセージ）に対するクライアントの応答が正しいかを確認する関数です。

SYNTAX

```
unsigned long __stdcall
_AhnHS_VerifyResponse (
    IN AHNHS_CLIENT_HANDLE hClient,
    IN unsigned char *pbyResponse,
    IN unsigned long nResponseLength
);
```

PARAMETERS

Parameter	Type	Description
hClient	AHNHS_CLIENT_HANDLE	クライアントハンドル
pbyResponse	char *	クライアントから受け取るデータのバッファー
nResponseLength	unsigned long	クライアントから受け取るデータの長さ

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

ERROR_ANTICPXSVR_INVALID_PARAMETER (Value = 0xE9040003)

内容: 入力に誤りがあります。

原因: hClient 、pbyResponse が NULL です。

対処方法: hClient 、pbyResponse が NULL でないか確認します。

ERROR_ANTICPXSVR_BAD_FORMAT (Value = 0xE9040004)

内容: フォーマットが不正です。

原因: 復号化処理で問題が発生しました。

対処方法: バッファをクライアントの_AhnHS_VerifyResponse 関数に正しく渡したか確認します。

ERROR_ANTICPXSVR_NO_WAITING (Value = 0xE9040006)

内容: 前回のリクエストメッセージに対して応答を受けませんでした。

原因: _AhnHS_MakeRequest 関数を呼び出したらクライアントから Ack メッセージを受信して _AhnHS_VerifyResponse 関数を呼び出した後、再度 _AhnHS_MakeRequest 関数を呼び出します。この同期に失敗する場合に発生するエラーです。

対処方法: クライアントセッションを確認して同期に問題が発生していないか確認します。

また、_AhnHS_VerifyResponse 関数の呼び出しが_AhnHS_MakeRequest 関数の呼び出し前に行われていないか確認します。

ERROR_ANTICPXSVR_NOT_ENOUGH_MEMORY (Value = 0xE9040007)

内容: メモリが不足しています。

原因: サーバーのメモリが不足しています。

対処方法: サーバー上でメモリリークが発生していない確認します。

ERROR_ANTICPXSVR_BAD_MESSAGE (Value = 0xE9040008)

内容: メッセージの暗号化または復号化に失敗しました。

原因: クライアントから返されたバッファの値が不正です。

対処方法: クライアントから返されたバッファが正常にpbyResponseへ入ったかを確認してください。

ERROR_ANTICPXSVR_REPLY_ATTACK (Value = 0xE9040009)

内容: パケット解析のための再送攻撃を検出しました。

原因: バッファが送信されましたが、HackShield は操作をしていません。

対処方法: ハッカーによるパケット解析攻撃の可能性があります。

ERROR_ANTICPXSVR_HSHIELD_FILE_ATTACK (Value = 0xE904000A)

内容: HackShield モジュールの改ざんを検出しました。

原因: HackShield ファイル EhSvc.dll が改ざんされました。

対処方法: クライアントの EhSvc.dll が改ざんされていないか確認してください。

ERROR_ANTICPXSVR_CLIENT_FILE_ATTACK (Value = 0xE904000B)

内容: クライアントファイルの改ざんを検出しました。

原因: ゲームクライアントのファイルが改ざんされました。

対処方法: ゲームクライアントが正常にアップデートされたかを確認してください。
また、ゲームクライアントがウイルスに感染されたかを確認してください。

ERROR_ANTICPXSVR_MEMORY_ATTACK (Value = 0xE904000C)

内容: メモリの改ざんを検出しました。

原因: クライアントのメモリが改ざんされた場合発生します。

対処方法: ハッキングツールによりメモリが改ざんされていないか確認してください。

ERROR_ANTICPXSVR_OLD_VERSION_CLIENT_EXPIRED (Value = 0xE904000D)

内容: 旧バージョンのクライアントが接続してきました。

原因: サーバーを停止しないで hsb ファイルをアップロードしようとした時に、既存の hsb ファイルを持つクライアントが接続を試みました。

対処方法: クライアントのバージョンを確認してください。クライアントのバージョンが低い場合は、パッチをあてて最新のバージョンに更新して下さい。

参考

HSBファイルを作成する時、HSBGen.iniの GrantOldSession = 0設定をした場合に発生する可能性があります。

または GrantOldSession = 1 設定をした場合に許可できるクライアントのバージョン数が MaxAllowedNumber を超過した場合発生します。

例)

[VERCT]

GrantOldSession = 1

MaxAllowedNumber = 1

現在、サーバーにアップロードされている HSBファイルと合うクライアントだけを許可する場合の設定です。

ERROR_ANTICPXSVR_UNKNOWN_CLIENT (Value = 0xE904000E)

内容: HSBファイルを作成する時に指定したクライアントのバージョンと合ってません。

原因: HSBGenでHSBファイルを作成する時のクライアントバージョンと実際クライアントのバージョンが違った場合発生します。

対処方法: 開発段階にエラーが発生した場合には再度HSBファイルを作成してテストを行ってください。サービス段階に特定ユーザーだけエラーが発生した場合にはクライアントファイルのパッチが正常に行われているか確認してください。また、ウイルスに感染していないか確認してください。

ERROR_ANTICPXSVR_NANOENGINE_FILE_ATTACK (Value = 0xE9040010)

内容: 3n.mhe ファイルの改ざんを検出しました。

原因: 3n.mhe ファイル が改ざんされました。

対処方法: 3n.mhe ファイル が改ざんされていないか確認してください。

ERROR_ANTICPXSVR_INVALID_HACKSHIELD_VERSION (Value = 0xE9040011)

内容: サーバーがサポートしないバージョンの HackShield です。

原因: クライアントの hshield.dat ファイルがサーバーにあるファイルより下位のバージョンのです。

対処方法: クライアントの hshield.dat ファイルとサーバーのファイルが同一か確認してください。

ERROR_ANTICPXSVR_INVALID_ENGINE_VERSION (Value = 0xE9040012)

内容: サーバーがサポートしないバージョンのヒューリスティックエンジンです。

原因: クライアントの 3n.mhe がサーバーにある 3n.mhe ファイルよりも低いバージョンです。

対処方法: 3n.mhe ファイルのバージョンを確認してください。

ERROR_ANTICPXSVR_VERIFY_EXCEPTION (Value = 0xE9040015)

内容: _AhnHS_VerifyResponse または _AhnHS_VerifyResponseEx 関数を実行中に例外(Exception)が発生しました。
(このエラーは、Windows OS でのみサポートします。)

原因: この例外は、多くの原因が考えられます。

対処方法: 例外が発生した場合、ミニダンプファイルを
‘VERIFYRESPONSE_year_month_date hour-minute-second.dmp’ というファイル名
で生成します。生成したダンプファイルをアンラボに送ってください。

ERROR_ANTICPXSVR_ABNORMAL_HACKSHIELD_STATUS (Value = 0xE9040018)

内容: HackShield の動作が正常ではありません。

原因: クライアントで HackShield が正常に動作していません。

対処方法: クライアントでハッキングツールが動作していないか確認してください。

ERROR_ANTICPXSVR_DETECT_CALLBACK_IS_NOTIFIED (Value = 0xE9040019)

内容: クライアントでハッキングツールが検出されました。

原因: クライアントでハッキングツールが検出されました。

対処方法: クライアントでハッキングツールが動作していないか確認してください。

ERROR_ANTICPXSVR_UNKNOWN (Value = 0xE90400FF)

内容: 定義されていないエラー

原因: 通常は発生しません。このエラーが発生した場合は、詳細に原因を特定する必要があります。

対処方法: アンラボにお問い合わせください。

注意

ERROR_ANTICPXSVR_BAD_MESSAGE
ERROR_ANTICPXSVR_REPLY_ATTACK
ERROR_ANTICPXSVR_HSHIELD_FILE_ATTACK
ERROR_ANTICPXSVR_CLIENT_FILE_ATTACK
ERROR_ANTICPXSVR_MEMORY_ATTACK
ERROR_ANTICPXSVR_OLD_VERSION_CLIENT_EXPIRED
ERROR_ANTICPXSVR_NANOENGINE_FILE_ATTACK
ERROR_ANTICPXSVR_UNKNOWN_CLIENT
ERROR_ANTICPXSVR_INVALID_HACKSHIELD_VERSION
ERROR_ANTICPXSVR_INVALID_ENGINE_VERSION
ERROR_ANTICPXSVR_VERIFY_EXCEPTION
ERROR_ANTICPXSVR_INVALID_PARAMETER
ERROR_ANTICPXSVR_ABNORMAL_HACKSHIELD_STATUS
ERROR_ANTICPXSVR_DETECT_CALLBACK_IS_NOTIFIED

□上記のエラーがリターンされた場合、クライアントとのセッションを切断してゲームを進むことができないように設定する必要があります。他のメッセージの場合はゲーム開発会社のポリシーによって追加可能です。

Example

_AhnHS_VerifyResponse 関数を呼び出したサンプルは次のとおりです。

サンプル

```
ulRet = _AhnHS_VerifyResponse ( hClient,  
stResponseBuf.byBuffer,  
stResponseBuf.nLength );  
  
if ( ulRet == ERROR_ANTICPXSVR_BAD_MESSAGE ||  
    ulRet == ERROR_ANTICPXSVR_REPLY_ATTACK ||  
    ulRet == ERROR_ANTICPXSVR_HSHIELD_FILE_ATTACK ||  
    ulRet == ERROR_ANTICPXSVR_CLIENT_FILE_ATTACK ||  
    ulRet == ERROR_ANTICPXSVR_MEMORY_ATTACK ||  
    ulRet == ERROR_ANTICPXSVR_OLD_VERSION_CLIENT_EXPIRED ||  
    ulRet == ERROR_ANTICPXSVR_NANOENGINE_FILE_ATTACK ||  
    ulRet == ERROR_ANTICPXSVR_UNKNOWN_CLIENT ||  
    ulRet == ERROR_ANTICPXSVR_INVALID_HACKSHIELD_VERSION ||  
    ulRet == ERROR_ANTICPXSVR_INVALID_ENGINE_VERSION ||
```

```
    ulRet == ERROR_ANTICPXSVR_VERIFY_EXCEPTION ||  
    ulRet == ERROR_ANTICPXSVR_INVALID_PARAMETER ||  
    ulRet == ERROR_ANTICPXSVR_ABNORMAL_HACKSHIELD_STATUS ||  
    ulRet == ERROR_ANTICPXSVR_DETECT_CALLBACK_IS_NOTIFIED )  
{  
    // エラー処理  
    // Client Out -> _AhnHS_CloseClientHandleを呼び出して  
    // hClient/ハンドルを閉める  
}
```

AhnHS_MakeResponse

DESCRIPTION

クライアントで使⽤します。サーバーから受け取った暗号化されたバージョン情報要求メッセージを復号化し、現在のクライアントファイルのバージョンを暗号化して応答メッセージを作成します。

SYNTAX

```
int __stdcall
_AhnHS_MakeResponse (
    unsigned char *pbyRequest,
    unsigned long ulRequestLength,
    PAHNHS_TRANS_BUFFER pResponseBuffer
);
```

PARAMETERS

Parameter	Type	Description
pbyRequest	unsigned char *	[IN] 要求メッセージバッファ
ulRequestLength	unsigned long	[IN] 要求メッセージの長さ
pResponseBuffer	PAHNHS_TRANS_BUFFER	[OUT] 応答メッセージバッファ

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

ERR_ANTICPXCNT_INVALID_PARAMETER (Value = 0xE4010001)

内容: パラメータの値が正しくありません。

原因: pbyRequest と pResponseBuffer が NULL の場合発生します。

対処方法: pbyRequest と pResponseBuffer の値を確認します。

ERR_ANTICPXCNT_INVALID_ADDRESS (Value = 0xE4010002)

内容: 不正メモリアドレスにアクセスしました。

原因: システムアーキテクチャの問題の可能性があります。

対処方法: アンラボにお問い合わせください。

ERR_ANTICPXCNT_NOT_ENOUGH_MEMORY (Value = 0xE40100013)

内容: メモリが不足しています。

原因: メモリアロケーションに失敗しました。

対処方法: サーバー上でメモリリークが発生していない確認します。

ERR_ANTICPXCNT_CRC_TABLE_INIT_FAILED (Value = 0xE4010004)

内容: 初期化に失敗しました。

原因: pbyRequest バッファサイズがANTICPX_TRANS_BUFFER_MAX (400)を超えています。

対処方法: ビルドに使用した HShield.h と ライブラリが最新バージョンか確認してください。pbyRequest バッファがサーバーから正常に受信されたか確認してください。

ERR_ANTICPXCNT_BAD_LENGTH (Value = 0xE4010005)

内容: メッセージサイズが不正です。

原因: メッセージバッファのサイズが正しくありません。

対処方法: HShield.h と ライブラリが最新バージョンか確認してください。

ERR_ANTICPXCNT_INSUFFICIENT_BUFFER (Value = 0xE4010006)

内容: バッファサイズが不正です。

原因: EhSvc.dll、HShield.lib と Hshield.h のバージョンが同一ではありません。

対処方法: HShield.h と ライブラリが最新バージョンか確認してください。

ERR_ANTICPXCNT_NOT_SUPPORTED (Value = 0xE4010007)

内容: このバージョンではサポートしていません。

原因: メッセージタイプがHackShieldから定義されなかった場合発生します。

対処方法: クライアントとサーバーの HackShield が同一バージョンか確認してください。

ERR_ANTICPXCNT_FILE_NOT_FOUND (Value = 0xE4010008)

内容: クライアントファイルが見つかりません。

原因: クライアントファイルが見つかりません。

対処方法: アンラボにお問い合わせください。

ERR_ANTICPXCNT_INVALID_MESSAGE_SIZE (Value = 0xE4010009)

内容: 入力メッセージのサイズが正しくありません。

原因: サーバーからのメッセージサイズが正しくなかった場合発生します。

対処方法: クライアントの HackShield とサーバーの HackShield のバージョンが同一か確認してください。

ERR_ANTICPXCNT_BAD_FORMAT (Value = 0xE401000A)

内容: フォーマットが不正です。

原因: サーバーからのメッセージサイズが正しくなかった場合発生します。

対処方法: クライアントの HackShield とサーバーの HackShield のバージョンが同一か確認してください。

ERR_ANTICPXCNT_DEBUGGER_DETECTED (Value = 0xE401000B)

内容: デバッガを検出しました。

原因: クライアントでデバッガー動作中です。

対処方法: デバッガを実行中か確認してください。

ERR_ANTICPXCNT_BAD_HSHIELD_MODULE (Value = 0xE401000C)

内容: HackShield モジュールのパスが不正です。または、HackShield モジュールが正しくありません。

原因: HackShield のモジュールがありません。

対処方法: HackShield モジュールのパスとモジュールが正しいものか確認します。

ERR_ANTICPX_CNT_BAD_CLIENT_FILE (Value = 0xE401000D)

内容: 不正なクライアントモジュールです。

原因: クライアントでの処理中に問題が発生しました。

ERR_ANTICPX_CNT_BAD_REQUEST (Value = 0xE401000E)

内容: サーバーから受け取った要求メッセージが正しくありません。

原因: サーバーから渡されたバッファが正しくなくて復号化失敗メッセージが発生しました。

対処方法: サーバから受信したバッファをチェックします。また、pbyRequest に正しいバッファの大きさが返ってきているか確認します。

ERR_ANTICPX_CNT_HSHIELD_CORE_ENGINE_NOT_WORKING (Value = 0xE401000F)

内容: HackShield コアエンジンが正常に動作していません。

原因: 通常は発生しません。このエラーが発生した場合は、詳細に原因を特定する必要があります。

対処方法: アンラボにお問い合わせください。

ERR_ANTICPX_CNT_UNKNOWN (Value = 0xE40100FF)

内容: ハッキング攻撃でシステムに問題が発生しました。。

原因: 通常は発生しません。このエラーが発生した場合は、詳細に原因を特定する必要があります。

対処方法: アンラボにお問い合わせください。

Example

_AhnHS_MakeResponse. 関数を呼び出したサンプルは次のとおりです。

サンプル

```
ulRet = _AhnHS_MakeResponse ( stRequestBuf.byBuffer,  
                             stRequestBuf.nLength,
```



```
&stResponseBuf );
```

```
if ( ulRet != ERROR_SUCCESS )  
{  
    // エラー処理  
}
```

5. サーバー連動クラック防止機能

5.1. 概要

AntiCPSvr は、2005年にアンラボで開発した「サーバー連動利用のファイル改ざん・メモリ改ざん検出SDK(Software Development Kit)」です。昨今のコンピューター技術の発達、とりわけハッキング技術の高度化により、より強力なクラック防止技術が必要になってきました。クライアント側だけではなく、サーバーと連動してマッチングすることで、より安全なクラックの防止が可能です。

機能

ファイル整合性のチェック

サーバーでAntiCpSvr.dll と AntiCpSvrFunc.h を適用し、クライアントではHackShieldモジュールであるHShield.lib と HShield.hを利用してサーバーとクライアント間で通信することでリアルタイムにファイル改ざん確認が可能です。

パケット整合性

パケットキャプチャし、特定パケットを作成して動作するようにするハッキングを防止するため、パケットの整合性を確認するモジュールが内部的に動作します。ネットワークでパケットのキャプチャや作成等を根本的に遮断する事ができます。ただし、この機能はサーバー連動クラック防止にかんするメッセージだけを対応します。

HackShield動作確認

サーバーでAntiCpSvr.dll と AntiCpSvrFunc.h を適用し、クライアントではHackShieldモジュールであるHShield.lib と HShield.hを利用してサーバーとクライアント間で通信することで簡単にHackShield動作の状態を確認することができます。それでサーバーはHackShieldが動作した後にクエリメッセージを送信する必要があります。

メモリ整合性機能

サーバーでAntiCpSvr.dll と AntiCpSvrFunc.h を適用し、クライアントではHackShieldモジュールであるHShield.lib と HShield.hを利用してサーバーとクライアント間で通信することでリアルタイムにゲームプロセスのメモリ改ざん確認が可能です。

HackShield エンジン整合性チェック

サーバーでAntiCpSvr.dll と AntiCpSvrFunc.h を適用し、クライアントではHackShield モジュールであるHShield.lib と HShield.hを利用してサーバーとクライアント間で通信することでエンジンファイル (v3warps.v3d と v3warps.v3d)整合性の確認ができます。

特徴

インターフェイス関数(API)の提供

インターフェイスDLL とライブラリは、開発者が簡単に AntiCPSvr の機能を利用できるように提供されます。提供されたインターフェイス DLL とライブラリを使用して開発者は、ゲームファイルの整合性をチェックして運用ステータスチェックすることができます。

ファイル情報作成プログラム

AntiCPSvrTool.exe は、ファイルデータとメモリ情報を保存するために提供されます。クライアントプログラムは、ファイル整合性、メモリ整合性を確認するためにサーバーとクライアント間で通信を行います。ゲーム開発者は、ファイルデータとメモリ整合情報を予めサーバー側に保存しておくことで整合性を確認することができます。
詳細は‘9.2.AntiCpSvr ツールの使い方’を参照して下さい。

テストプログラムの提供

テストプログラム Amazon.exe は、AntiCPSvr で提供する API を利用して実装されています。Amazon.exe は、HackShield と AntiCPSvr のテスト機能を実装しています。

システムアーキテクチャ

AntiCPSvr は、AntiCpSvr.dll とHShield.lib、EhSvc.dll を提供するします。DLL とライブラリをサーバーとクライアントに適用して利用します。AntiCPSvr のアーキテクチャと動作原理は以下のとおり:

AntiCPSvr.dll (インターフェース dll)

サーバーで使用します。サーバーで要求メッセージを生成し、クライアントから受け取ったメッセージを使ってクライアントのファイルやメモリが正常に動作しているかを確認するAPI を提供します。

HShield.lib (HackShield ライブラリ)

クライアントで使⽤します。サーバーから転送された要求メッセージを受けて要求された内容を実⾏し、結果として得た各種情報を暗号化してサーバーに送るデータを生成する API を提供します。

AntiCpSvrTool.exe (ファイル情報生成プログラム)

サーバーで使⽤するファイル情報やメモリ情報のファイルを生成してゲームファイルの完全性の有無や実⾏するゲームのメモリの完全性の有無を判断できるようにします。ファイル情報データは生成する度に異なるため、ゲームサーバーが複数であればサーバーごとに適⽤することで高いセキュリティを保つことができます。メモリ情報データは AntiCpSvrTool.exe を実⾏した状態でゲームクライアントを実⾏したら自動で作成されます。

HackShield.crc (クライアントCRCファイル)

AntiCpSvrTool.exe で作成されます。クライアントの整合性を確認するために使⽤されます。

HSPub.key (サーバー連動認証キーファイル)

アクセスしてきたクライアントの HackShield を認証するために使⽤します。HackShield.crc ファイルと同一フォルダに配置する必要があります。

3n.mhe (ヒューリスティックエンジンファイル)

クライアントで使⽤されるエンジンファイルです。HackShield.crc ファイルと同じフォルダに配置されるとクライアントでは、旧バージョンの 3n.mhe ファイルとの接続を切断します。

hshield.dat (HackShield バージョンファイル)

クライアントで使⽤されるデータファイルです。HackShield.crc ファイルと同じフォルダに配置されるとクライアントでは、旧バージョンの hshield.dat ファイルとの接続を切断します。

参考

サーバーの HackShield.crc ファイルフォルダパスに 3n.mhe または hshield.dat ファイルがある場合、クライアント側では旧バージョンのファイルの接続を切断します。(3n.mhe または hshield.dat をアップロードした時にサーバーをリスタートする必要はありません。)

“Disconnected” の状態は下の表ようになります。

_AntiCpSvr_AnalyzeAckMsg 関数から以下のエラーを返します。

ERROR_ANTICPSVR_ANALGUIDACKMSG_HSHIELDDENIED_NEWSESSION

ERROR_ANTICPSVR_ANALGUIDACKMSG_NANODENIED_NEWSESSION

エラーを返すとクライアントコネクションを切断してください。

表 5-1 サーバー連動バージョン管理

File name	Version	Connection
3n.mhe	Client < Server	Disconnected
	Client ≥ Server	Connected
hshield.dat	Client < Server	Disconnected
	Client ≥ Server	Connected

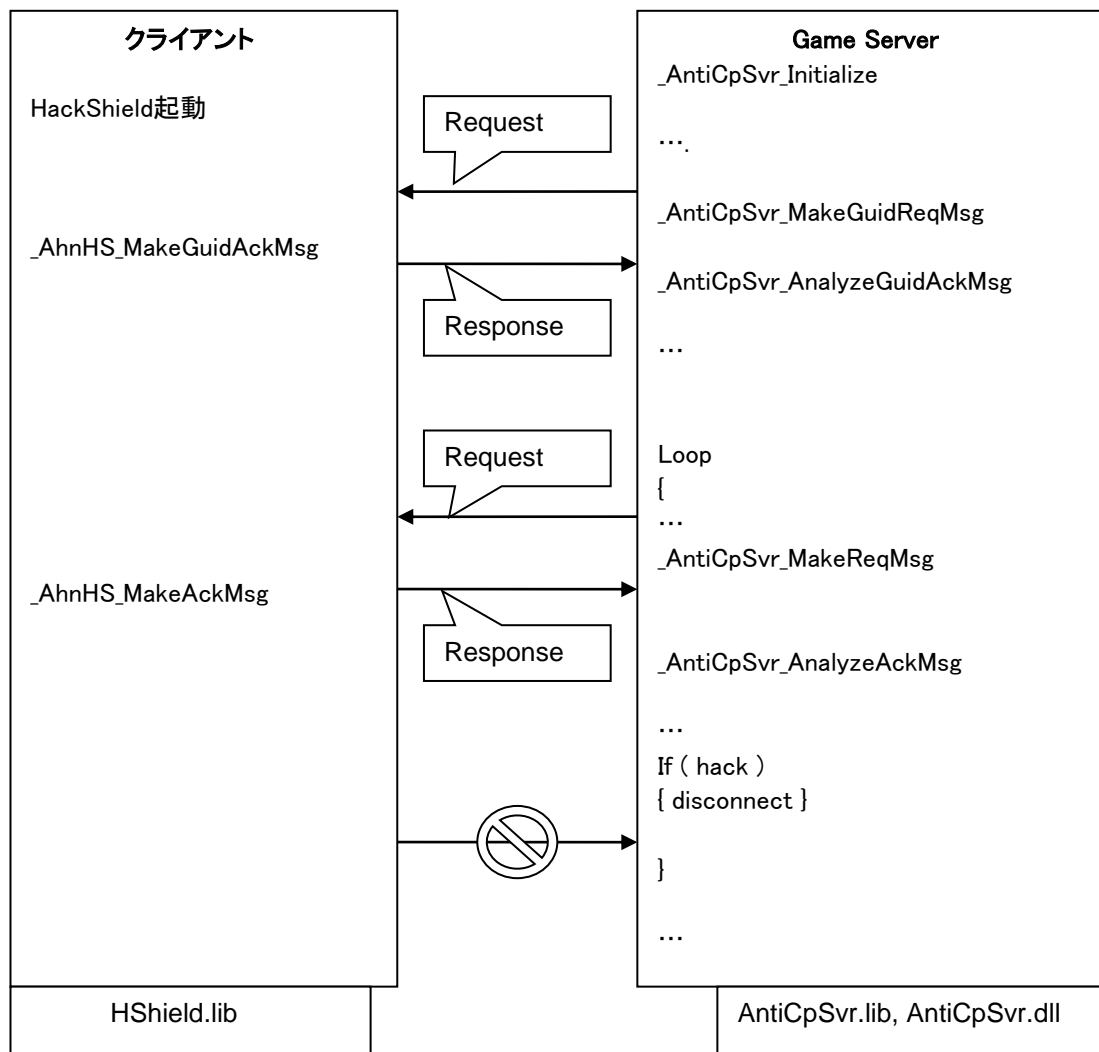


図. 5-1 AntiCpSvr の動作

ゲームクライアントがサーバーに接続すると、そのクライアントがどのようなバージョンの実行ファイルを使用しているのかを知るために GUID 要求メッセージを送り、それに対する応答メッセージを受け取ります。このとき受け取ったバージョン情報がサーバーで許可しているバージョンであるか _AntiCpSvr_AnalyzeGuidAckMsg API により知ることができます。もし許可しているバージョンであれば接続が維持されますが、許可されていないバージョンであれば接続を切断することで最新パッチを適用せずにゲームをする行為を遮断することができます。

ゲームサーバーではゲーム開発会社のポリシーに従って一定間隔で要求メッセージと応答メッセージを確認し、クライアントの実行ファイルやメモリ、HackShield モジュール、エンジンに対するクラックの有無を確認することができます。応答メッセージをゲームサーバーに転送できない場合もハッキングと判断します。

5.2. Application Programming

AntiCPSvr の API を使ってファイルやメモリの完全性確認を実行する方法について説明します。

参考

本書で使用しているサンプルコードは Microsoft Visual C++ 6.0 を対象とした C/C++ 言語により作成されています。プログラミングに使われている言語は各プログラムの特性やシステム環境によって変更することができます。

プログラミング適用方法

AntiCPSvr を使ってプログラミングを始める前に次のような準備作業を行います。:

5.2.1.1.AntiCPSvr関連ファイル

AntiCPSvr関連ファイル

表 5-2 AntiCPSvr関連ファイル

File name	Installation folder	Description
AntiCPSvrfunc.h	[プログラムソースフォルダ]	サーバー用ヘッダーファイル
AntiCPSvr.dll	[プログラムソースフォルダ]	サーバー用DLLファイル
AntiCPSvr.lib	[プログラムソースフォルダ]	AntiCPSvr.dllのインポートライブラリ
HackShield.crc	[プログラム実行フォルダ]	クライアントの整合性検証ファイル
HSPub.key	[プログラム実行フォルダ]	サーバー連動認証キーファイル
3n.mhe	[CRC ファイルと同一フォルダ]	Heuristic エンジンバージョン管理ファイル
hshield.dat	[CRC ファイルと同一フォルダ]	HackShield version 管理ファイル
HShield.h	[プログラムソースフォルダ]	クライアント用ヘッダーファイル
HShield.lib	[プログラムソースフォルダ]	クライアント用ライブラリファイル
EhSvc.dll	[プログラムソースフォルダ]	クライアント用 dll ファイル

5.2.1.2.適用方法

サーバー適用方法

1. AntiCPSvrTool.exeを利用してHackShield.crc ファイルを作成します。
(詳細は、[9.2.AntiCpSvr ツール](#) を参照してください。)
2. HSPub.key を HackShield.crc ファイルがあるフォルダにコピーします。
3. 3n.mhe と hshield.dat を CRC ファイルがあるフォルダにコピーします。
(これは、必須ではありませんが HackShield のバージョン管理の観点からお勧めします。)

参考

サーバーにCRCファイルがあるフォルダにクライアントの3n.mheとhshield.datファイルをコピーすることで旧バージョンのファイルを使う3n.mhe and hsクライアント場合、接続できないように管理することができます。

4. AntiCpSvr.dllのインポートライブラリ(AntiCpSvr.lib) をプロジェクトに追加します。
5. AntiCPSvrFunc.h ファイルをソースにインクルードします。
6. _AntiCpSvr_Initialize を呼び出して初期化作業を実行します。
7. クライアント接続時に _AntiCpSvr_MakeGuidReqMsg を呼び出してクライアントに伝える暗号化されたバージョン情報要求メッセージを生成します。
8. 暗号化されたバージョン情報要求メッセージをクライアントに転送します。
9. クライアントで暗号化されたバージョン情報応答メッセージを受け取ります。
10. _AntiCpSvr_AnalyzeGuidAckMsg を呼び出してクライアントから受け取った暗号化されたバージョン情報応答メッセージを分析します。ERROR_SUCCESS が表示されなければ接続を切ります。
11. _AntiCpSvr_MakeReqMsg を呼び出してクライアントに伝える暗号化された要求メッセージを生成します。
12. 暗号化された要求メッセージをクライアントに転送します。
13. クライアントで暗号化された応答メッセージを受け取ります。
14. _AntiCpSvr_AnalyzeAckMsg を呼び出してクライアントで受け取った暗号化された応答メッセージを分析します。ERROR_SUCCESS が表示されなければ接続を切ります。
15. サーバープログラム終了時に _AntiCpSvr_Finalize 関数を呼び出します。

クライアント適用方法

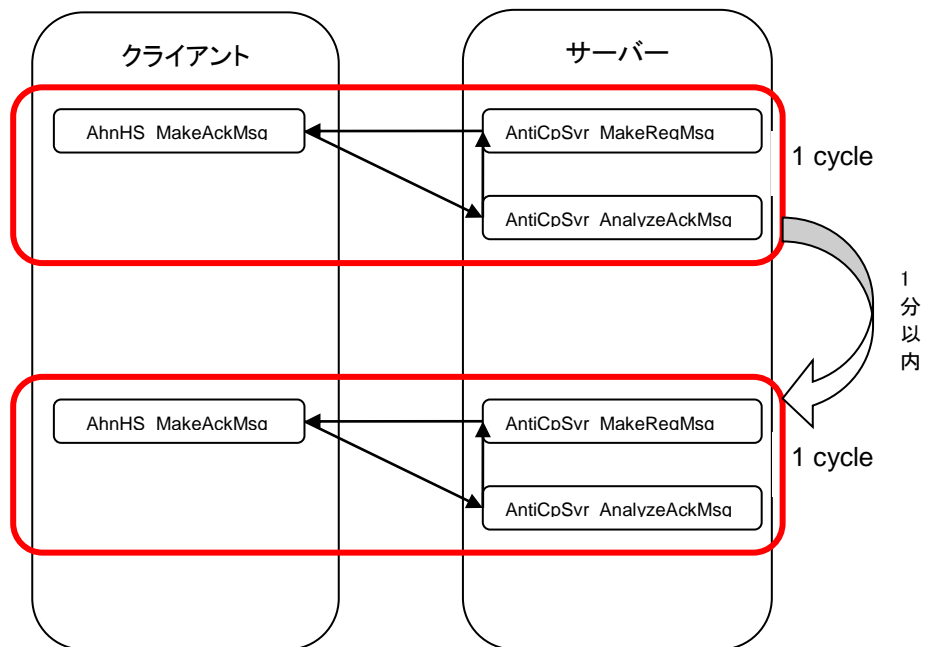
1. HShield.lib ファイルを作業するプロジェクトに含めます。
2. 提供されている HShield.h ファイルを作業するソースファイル内に置きます。
3. サーバーから暗号化されたバージョン情報要求メッセージを受け取ります。
4. _AhnHS_MakeGuidAckMsg を呼び出してサーバーに送る暗号化されたバージョン情報応答メッセージを生成します。
5. サーバーから暗号化された要求メッセージを受け取ります。

6. `_AhnHS_MakeAckMsg` を呼び出してサーバーに送る暗号化された応答メッセージを生成します。
7. 応答メッセージをサーバーに転送します。

サーバー連動サイクル

説明: 1サイクルは以下のような流れで構成されます。:
サーバーがリクエストメッセージを送信します。
クライアントは応答メッセージを作成し、サーバーへ送信します。
その結果サーバーは、レスポンスメッセージを受信します。

サイクル: 1 サイクルが 1 分以内に終了することを推奨します。
(ゲームの特性によりもっと長いサイクルタイムが必要な場合があります。しかし、その分ハッキングに対する危険性も増します。)



5.3. Application Programming Interface

AntiCpSvr Initialize

DESCRIPTION

ゲーム実行ファイル、メモリ、HackShield モジュール、エンジンファイルに関する情報を持つデータをロードしてその他の初期化作業を実行します。HackShield.crc と HSPub.key ファイルが lpszHashFilePath で指定するパスに存在する必要があります。

SYNTAX

```
unsigned long _stdcall
_AntiCpSvr_Initialize (
    IN const char *lpszHashFilePath
);
```

PARAMETERS

Parameter	Value	Description
lpszHashFilePath	const char *	CRCデータファイル(HackShield.crc)があるフォルダのフルパス

RETURN VALUE

ERROR_SUCCESS . (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

ERROR_ANTICPSVR_INIT_INVALIDPARAM (Value = 0x1C001)

内容: 入力データが不正です。

原因: lpszHashFilePath が null です。

対処方法: lpszHashFilePath を確認してください。

ERROR_ANTICPSVR_INIT_INSERTCRCDATATOLIST_FAIL (Value = 0x1C002)

内容: CRC ファイルが追加できません。

原因: CRC ファイルが追加できません、またはメモリが不足しています。

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_INIT_READPUBLICKEY_FAIL (Value = 1C004)

内容: HSPub.key がみつからないか、破損しています。

原因: HSPub.key が存在しないか、破損しています。

対処方法: HSPub.key が存在するか確認してください。

(Windows サーバーの場合、“¥”を“/”の代わりに使用してください。)

HSPub.key を コピーしなおしてください。

Example

_AntiCpSvr_Initialize 関数を呼び出したサンプルは次のとおりです。

サンプル

```
dwRet = _AntiCpSvr_Initialize ( m_strFileCrcDataPath    // [in]
);

if( dwRet != ERROR_SUCCESS)
{
    // エラー処理
}
```

_AntiCpSvr_Finalize

DESCRIPTION

動的に割り当てられたメモリを開放して内部的に使用したデータに対する Clean Up 過程を実行します。

SYNTAX

```
void __stdcall  
_AntiCpSvr_Finalize( )
```

PARAMETERS

None.

RETURN VALUE

None.

Example

_AntiCpSvr_Finalize関数を呼び出したサンプルは次のとおりです。

サンプル

```
_AntiCpSvr_Finalize ();
```

_AntiCpSvr_MakeGuidReqMsg

DESCRIPTION

クライアントに送る暗号化されたバージョン情報要求メッセージを生成します。

SYNTAX

```
unsigned long _stdcall  
_AntiCpSvr_MakeGuidReqMsg (  
    OUT unsigned char *pbyGuidReqMsg,  
    OUT unsigned char *pbyGuidReqInfo  
);
```

PARAMETERS

Parameter	Type	Description
pbyGuidReqMsg	unsigned char *	クライアントに送る暗号化されたバージョン情報要求メッセージであり、サイズは AntiCpSvrFunc.h に定義された SIZEOF_GUIDREQMSG です。
pbyGuidReqInfo	unsigned char *	_AntiCpSvr_AnalyzeGuidAckMsg 関数で分析する時に必要なバージョン要求メッセージ情報であり、バッファのサイズは AntiCpSvrFunc.h に定義された SIZEOF_GUIDREQINFO です。

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

ERROR_ANTICPSVR_MAKEGUIDREQMSG_INVALIDPARAM (Value = 0x1C040)

内容: 入力データに不正があります。

原因: pbyGuidReqMsg と pbyGuidReqInfo が NULL です。

対処方法: pbyGuidReqMsg と pbyGuidReqInfo が NULL でないか確認してください。

ERROR_ANTICPSVR_MAKEGUIDREQMSG_MAKESKEY_FAIL (Value = 0x1C041)

内容: メッセージの暗号化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_MAKEGUIDREQMSG_INITCRYPT_FAIL (Value = 0x1C042)

内容: 暗号化/復号化の初期化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_MAKEGUIDREQMSG_ENCRYPT_FAIL (Value = 0x1C043)

内容: 暗号化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

Example

_AntiCpSvr_MakeGuidReqMsg関数を呼び出したサンプルは次のとおりです。

```
サンプル

dwRet = _AntiCpSvr_MakeGuidReqMsg (
byGuidReqMsg, // [out]
    byGuidReqInfo // [out]
);

if( dwRet != ERROR_SUCCESS)
{
    // エラー処理
}
```

_AntiCpSvr_AnalyzeGuidAckMsg

DESCRIPTION

クライアントから送信した暗号化されたバージョン応答メッセージを復号化し、現在使用しているクライアントのバージョンがサーバーで許可しているバージョンであるか確認します。

SYNTAX

```
unsigned long _stdcall
_AntiCpSvr_AnalyzeGuidAckMsg (
    IN unsigned char *pbyGuidAckMsg,
    IN unsigned char *pbyGuidReqInfo,
    OUT PHSHIELD_CLIENT_CONTEXT pCrcInfo
);
```

PARAMETERS

Parameter	Type	Description
pbyAckMsg	unsigned char *	クライアントから送信した暗号化されたバージョン応答メッセージ
pbyGuidReqInfo	unsigned char *	_AntiCpSvr_MakeGuidReqMsg 関数から生成したバージョン応答メッセージの情報
pCrcInfo	PHSHIELD_CLIENT_CONTEXT	クライアントが使うCRC情報に対する構造体ポインター

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

ERROR_ANTICPSVR_ANALGUIDACKMSG_INVALIDPARAM (Value = 0x1c050)

内容: 入力データに不正があります。

原因: pbyGuidReqMsg と pbyGuidReqInfo、pCrcInfo が NULL です。

対処方法: pbyGuidReqMsg と pbyGuidReqInfo、pCrcInfo が NULL でないか確認してください。(CrcInfo 構造体の宣言と、リターンアドレス (&CecInfo) を確認してください。)

ERROR_ANTICPSVR_MAKEGUIDREQMSG_MAKESKEY_FAIL (Value = 0x1C041)

内容: メッセージの暗号化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_MAKEGUIDREQMSG_INITCRYPT_FAIL (Value = 0x1C042)

内容: 暗号化/復号化の初期化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_ANALGUIDACKMSG_DECRYPT_FAIL (Value = 0x1c053)

内容: メッセージの復号に失敗しました。

原因: メッセージを正しく受信できませんでした。

対処方法: クライアントから送信されたバッファとそのサイズが上記の関数の入力バッファ(pbyAckMsg)に正常に受けられたか確認してください。通常、クライアントから送信する時に入力バッファを暗号化します。正常に暗号化したバッファだけが復号することができます。暗号化に失敗している場合、復号に失敗します。

ERROR_ANTICPSVR_ANALGUIDACKMSG_PACKET_ERROR (Value = 0x1c054)

内容: パケットエラーが発生しました。

原因: サーバーから MakeGuidReqMsg 関数を呼び出してクライアントで MakeGuidAckMsg 関数を使用して応答する必要があります。もし同期が失敗する場合、トラブルが発生している可能性があります。

対処方法: サーバーとクライアントは常に同期している必要があります。パケットがどこでロストしているかネットワークトラブルが発生していないか確認してください。

ERROR_ANTICPSVR_ANALGUIDACKMSG_DENIED_NEWSESSION (Value = 0x1c055)

内容: 既存セッションは許可されていますが、新規セッションは許可されていません。

原因: サーバーで使っているCRCを作成したクライアントのバージョンではなく、旧バージョンのクライアントが接続する時リターンされるエラーです。

対処方法: エラーが開発フェイズで発生した場合は、CRC を新たに作成してサーバーをリスタートしてください。サービス運用中にエラーが発生した場合は、クライアントのファイルが正しくパッチされているか、ウイルスに感染していないか確認してください。

ERROR_ANTICPSVR_ANALGUIDACKMSG_GETGUIDFROMCRCFILE_ERROR (Value = 0x1c056)

内容: CRC の読み込みに失敗しました。

原因: サーバーにHackShield.crc ファイルが存在しないか破損しています。

対処方法: サーバーにHackShield.crcファイルが存在しているか確認して、新たにCRCファイルを作成してください。

ERROR_ANTICPSVR_ANALGUIDACKMSG_INSERTCRCDATATOLIST_FAIL (Value = 0x1c057)

内容: CRC データの更新に失敗しました。

原因: サーバーメモリの不足により発生する可能性があります。

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_ANALGUIDACKMSG_INVALIDGUID (Value = 0x1c058)

内容: サポートしないバージョンです。

原因: サーバーが許可しないバージョンのクライアントです。

対処方法: エラーが開発フェイズで発生した場合は、CRC を新たに作成してサーバーをリスタートしてください。サービス運用中にエラーが発生した場合は、クライアントのファイルが正しくパッチされているか、ウイルスに感染していないか確認してください。

ERROR_ANTICPSVR_ANALGUIDACKMSG_HSHIELDDENIED_NEWSESSION (Value = 0x1c059)

内容: サーバーがサポートしない HackShield のバージョンです。

原因: クライアントのHShield.datファイルバージョンがサーバーからサポートしないバージョンです。このエラーが発生した場合、クライアントのhshield.dat ファイルがサーバーより古くないか確認してください。

対処方法: クライアントのHackShieldアップデートが正しくされていないか、ハッキングが行われていないか確認してください。

**ERROR_ANTICPSVR_ANALGUIDACKMSG_INSERTHSHIELDCRCDATATOLIST_FAIL
(Value = 0x1c05B)**

内容: HackShield バージョンの更新に失敗しました。

原因: サーバーで HackShield バージョンの更新に失敗しました。

対処方法: アンラボにお問い合わせください。

**ERROR_ANTICPSVR_ANALGUIDACKMSG_NANODENIED_NEWSESSION
(Value = 0x1c05D)**

内容: サーバーがサポートしない エンジン (3n.mhe) のバージョンです。

原因: サーバーがサポートしないバージョンのエンジンファイルです。このエラーが発生した場合、クライアントの 3n.mhe ファイルがサーバーより古くないか確認してください。

対処方法: クライアントが正しくアップデートされているか、ハッキングが行われていないか確認してください。

**ERROR_ANTICPSVR_ANALGUIDACKMSG_INSERTNANOCRCDATATOLIST_FAIL
(Value = 0x1c05E)**

内容: 3n.mhe ファイルバージョンの更新に失敗しました。

原因: サーバーで3n.mhe ファイルバージョンの更新に失敗しました。

対処方法: アンラボにお問い合わせください。

**ERROR_ANTICPSVR_ANALGUIDACKMSG_CLIENTNANOENGINE_OLDVERSION
(Value = 0x1c05F)**

内容: サーバーがサポートしない HackShield のバージョンです。

原因: サーバーがサポートしない HShield.dat のバージョンです。

対処方法: hshield.dat ファイルが改ざんされていないか確認してください。

注意

上記エラーが返ってきた場合、直ちにクライアントからの接続を切断し、ゲームを終了することを推奨します。

Example

_AntiCpSvr_AnalyzeGuidAckMsg関数を呼び出したサンプルは次のとおりです。

Example

```
HSIELD_CLIENT_CONTEXT CrcInfo = {0, };

dwRet = _AntiCpSvr_AnalyzeGuidAckMsg ( byGuidAckMsg, // [in]
byGuidReqInfo, // [in]
&CrcInfo // [out]
);

if( dwRet != ERROR_SUCCESS)
{
    // エラー処理
}
```

出力パラメータにリターンされるCRC情報に対する構造体(CrcInfo)は、クライアントがセッションを切断するまでに使われる情報ですので、クライアント別の保存が必要ですし、_AntiCpSvr_MakeReqMsg 関数と _AntiCpSvr_AnalyzeAckMsg 関数の入力パラメータとして利用されます。

AntiCpSvr_MakeReqMsg

DESCRIPTION

クライアントに送る暗号化された CRC 要求メッセージを生成します。

SYNTAX

```
DWORD _stdcall  
_AntiCpSvr_MakeReqMsg (  
    IN PHSHIELD_CLIENT_CONTEXT pCrcInfo,  
    OUT unsigned char *pbyReqMsg,  
    OUT unsigned char *pbyReqInfo,  
    IN unsigned long ulOption  
);
```

PARAMETERS

Parameter	Type	Description
pCrcInfo	PHSHIELD_CLIENT_CONTEXT	_AntiCpSvr_AnalyzeGuidAckMsg関数からリターンされたクライアントが使うCRC情報に対する構造体ポインタ
pbyReqMsg	unsigned char *	クライアントに送信する暗号化されたCRC要求メッセージ、バッファのサイズはAntiCpSvrFunc.hに定義されたSIZEOF_REQMSGです。
pbyReqInfo	unsigned char *	_AntiCpSvr_AnalyzeAckMsg関数で分析する時に必要なCRC要求メッセージ情報でバッファのサイズはAntiCpSvrFunc.hに定義されたSIZEOF_REQINFOです。
ulOption	unsigned long	_AntiCpSvr_AnalyzeAckMsg関数で分析する時にどんな情報に対するRequest Messageを作成するか決めるFlagであり、AntiCpSvrFunc.hに下記のように定義されています。 ANTICPSVR_CHECK_GAME_MEMORY, .ANTICPSVR_CHECK_HACKSHIELD_FILE, ANTICPSVR_CHECK_GAME_FILE, ANTICPSVR_CHECK_NANOENGINE_FILE, and ANTICPSVR_CHECK_ALL. ただし、安全のために最初呼び出す時にはANTICPSVR_CHECK_ALLオプションを使って全体に対する安全有無を確認し、その後からはperformanceのためにANTICPSVR_CHECK_GAME_MEMORYオプションだけを利用することを勧めます。

RETURN VALUE

ERROR_SUCCESS

内容: 関数の呼び出しに成功した時に返される値です。

ERROR_ANTICPSVR_MAKEREQMSG_INVALIDPARAM (Value = 0x1C010)

内容: 入力値が正しくありません。

原因: pbyGuidReqMsg と pbyGuidReqInfo、pCrcInfo が NULL です。

対処方法: uloption が “0” または、入力パラメータが NULL でないか確認してください。

ERROR_ANTICPSVR_MAKEREQMSG_MAKESKEY_FAIL (Value = 0x1C011)

内容: メッセージの暗号化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_MAKEREQMSG_INITCRYPT_FAIL (Value = 0x1C012)

内容: 暗号化/復号化の初期化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_MAKEREQMSG_ENCRYPT_FAIL (Value = 0x1C013)

内容: メッセージの復号に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_MAKEREQMSG_GETRNDHASHINFO_FAIL (Value = 0x1C014)

内容: CRC データの取得に失敗しました。.

原因: pCrcInfo が NULL か 不正な値です。

対処方法:

Example

_AntiCpSvr_MakeReqMsg関数を呼び出したサンプルは次のとおりです。

サンプル

```
// byClientInfo、CrcInfoはGUID要求した結果の値であり、  
// クライアントの接続が切断されるまでは値が続きます。  
  
dwRet = _AntiCpSvr_MakeReqMsg ( &CrcInfo,                // [in]  
                                byReqMsg,                 // [out]  
                                byReqInfo,                // [out]  
                                ANTICPSVR_CHECK_ALL       // [in]  
                                );  
  
if( dwRet != ERROR_SUCCESS)  
{  
    // エラー処理  
}
```

OUT で返される byReqInfo は該当する要求メッセージに対する応答メッセージを分析する時に使用されます。分析するには _AntiCpSvr_AnalyzeAckMsg 関数を呼び出す前に情報を保存し、安全なマルチスレッド処理を行う必要があります。

_AntiCpSvr_AnalyzeAckMsg

DESCRIPTION

クライアントから受け取った暗号化された CRC 応答メッセージを復号化してゲームファイルやパケットの完全性や HackShield が正常に動作しているかを確認したり、HackShield モジュールの完全性やメモリの完全性を確認します。

SYNTAX

```
unsigned long _stdcall  
_AntiCpSvr_AnalyzeAckMsg (  
    IN PHSIELD_CLIENT_CONTEXT pCrcInfo,  
    IN unsigned char *pbyAckMsg,  
    IN unsigned char *pbyReqInfo  
);
```

PARAMETERS

Parameter	Type	Description
pCrcInfo	PHSHIELD_CLIENT_CONTEXT	_AntiCpSvr_AnalyzeGuidAckMsg関数からリターンされた該当クライアントが使うCRC情報に対する構造体ポインタ
pbyAckMsg	unsigned char *	クライアントから送信したCRC応答メッセージ
pbyReqInfo	unsigned char *	_AntiCpSvr_MakeReqMsg関数からリターンされたCRC要求メッセージ情報

RETURN VALUE

ERROR_SUCCESS (Value = 0x000000)

内容: 関数の呼び出しに成功した時に返される値です。

原因: Normal

ERROR_ANTICPSVR_ANALACKMSG_INVALIDPARAM (Value = 0x1C020)

内容: 入力データが正しくありません。

原因: pbyGuidReqMsg と pbyGuidReqInfo、pCrcInfo が NULL です。

対処方法: pbyGuidReqMsg と pbyGuidReqInfo、pCrcInfo が NULL でないか確認してください。

ERROR_ANTICPSVR_ANALACKMSG_MAKESKEY_FAIL (Value = 0x1C021)

内容: メッセージの暗号化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_ANALACKMSG_INITCRYPT_FAIL (Value = 0x1C022)

内容: 暗号化/復号化の初期化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

ERROR_ANTICPSVR_ANALACKMSG_DECRYPT_FAIL (Value = 0x1C023)

内容: メッセージの復号に失敗しました。

原因: メッセージを正しく受信できませんでした。

対処方法: クライアントから送信されたバッファとそのサイズが上記の関数の入力バッファ(pbyAckMsg)に正常に受けられたか確認してください。通常、クライアントから送信する時に入力バッファを暗号化します。正常に暗号化したバッファだけが復号することができます。暗号化に失敗している場合、復号に失敗します。

ERROR_ANTICPSVR_ANALACKMSG_HSHIELD_ERROR (Value = 0x1C024)

内容: HackShield モジュールが改ざんされました。

原因: EhSvc.dll と hshield.dat はいつも同一バージョンが配布されます。このエラーが発生した場合、HackShield モジュールが改ざんされている可能性があります。

対処方法: クライアントの HackShield ファイルが正しいものか確認してください。また、正常にパッチされているか確認してください。ハッキング攻撃でHackShield モジュールが改ざんされている可能性があります。

ERROR_ANTICPSVR_ANALACKMSG_PACKET_ERROR (Value = 0x1c025)

内容: パケットエラーが発生しました。

原因: サーバーからMakeReqMsg 関数を呼び出してクライアントでMakeAckMsg関数を使用して応答する必要があります。もし同期が失敗する場合、トラブルが発生している可能性があります。

対処方法: サーバーとクライアントは常に同期している必要があります。パケットがどこでロストしているかネットワークトラブルが発生していないか確認してください。

ERROR_ANTICPSVR_ANALACKMSG_FILECRC_ERROR ERROR (Value = 0x1c026)

内容: クライアントモジュールの改ざんを検出しました。

原因: クライアントファイルが改ざんされました。

対処方法: クライアントが正しくアップデートされているか、ハッキングが行われていないか確認してください。

ERROR_ANTICPSVR_ANALACKMSG_MEMORYCRC_ERROR (Value = 0x1c027)

内容: メモリの改ざんを検出しました。

原因: メモリが改ざんされました。

対処方法:

ERROR_ANTICPSVR_ANALACKMSG_INVALIDSESSION_ERROR (Value = 0x1c028)

内容: サーバーがサポートしない HackShield バージョンのクライアントです。

原因: パッチのインストール時にセッションを切断するように設定した場合、発生します。

対処方法: このエラーは、CRC ファイルをアップロードしたあとでサーバーを再起動しないと発生する可能性があります。(CRC ファイルを作成した時、旧バージョンのクライアントを許可していないか確認してください。)

ERROR_ANTICPSVR_ANALACKMSG_NANOENGINECRC_ERROR (Value = 0x1c02B)

内容: 3n.mhe ファイルの改ざんを検出しました。

原因: 3n.mhe ファイルの改ざんを検出しました。

対処方法: クライアントが正しくアップデートされているか、改ざんが行われていないか確認してください。

注意

上記エラーが返ってきた場合、直ちにクライアントからの接続を切断し、ゲームを終了することを推奨します。

Example

_AntiCpSvr_AnalyzeAckMsg関数を呼び出したサンプルは次のとおりです。

サンプル

```
// byClientInfo、CrcInfoはMakeReqMsgを要求する時の値であり、  
// クライアントの接続が切断されるまでは値が続きます。  
  
dwRet = _AntiCpSvr_AnalyzeAckMsg ( &CrcInfo,    // [in]  
                                   byAckMsg,      // [in]  
                                   byReqInfo      // [in]  
                                   );  
  
if( dwRet != ERROR_SUCCESS)  
{  
    // エラー処理  
}
```

AhnHS_MakeGuidAckMsg

DESCRIPTION

クライアントで使⽤します。サーバーから受け取った暗号化されたバージョン情報要求メッセージを復号化し、現在のクライアントファイルのバージョンを暗号化して応答メッセージを生成します。

SYNTAX

```
int __stdcall
_AhnHS_MakeGuidAckMsg (
    IN unsigned char *pbyGuidReqMsg,
    OUT unsigned char *pbyGuidAckMsg
);
```

PARAMETERS

Parameter	Type	Description
pbyGuidReqMs g	unsigned char *	サーバーから送信したバージョン要求メッセージでバッファサイズはHShield.hに定義されたSIZEOF_GUIDREQMSGです。
pbyGuidAckMs g	unsigned char *	サーバーに送信する暗号化されたバージョン応答メッセージでバッファのサイズはHShield.hに定義されたSIZEOF_GUIDACKMSGです。

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

原因: Normal

対処方法:

ERROR_ANTICPNT_MAKEGUIDACKMSG_INVALIDPARAM (Value = 0x10010)

内容: 入力値が正しくありません。

原因: pbyGuidReqMsg と pbyGuidAckMsg が NULL です。

対処方法:

ERROR_ANTICPCNT_MAKEGUIDACKMSG_INITCRYPT_FAIL (Value = 0x10012)

内容: 暗号化/復号化の初期化に失敗しました。

ERROR_ANTICPCNT_MAKEGUIDACKMSG_DECRYPTMESSAGE_FAIL (Value = 0x10013)

内容: メッセージの復号に失敗しました。

原因: メッセージを正しく受信できませんでした。

対処方法: クライアントから送信されたバッファとそのサイズが上記の関数の入力バッファ(pbyAckMsg)に正常に受けられたか確認してください。通常、クライアントから送信する時に入力バッファを暗号化します。正常に暗号化したバッファだけが復号することができます。暗号化に失敗している場合、復号に失敗します。

ERROR_ANTICPCNT_MAKEGUIDACKMSG_GETIDENTIFIER_FAIL (Value = 0x10014)

内容: GUID値が見つかりません。

原因: GUIDを生成する途中にHackShield モジュールまたはゲームクライアントファイルに問題が発生しました。

対処方法: HackShield モジュールとゲームクライアントファイルが正常か確認してください。

ERROR_ANTICPCNT_MAKEGUIDACKMSG_ENCRYPTMESSAGE_FAIL (Value = 0x10016)

内容: メッセージの暗号化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

Example

_AhnHS_MakeGuidAckMsg関数を呼び出したサンプルは次のとおりです。

サンプル

```
dwRet = _AhnHS_MakeGuidAckMsg( byGuidReqMsg,          // [in]
```

```
byGuidAckMsg // [out]
);

if( dwRet != ERROR_SUCCESS)
{
    // エラー処理
}
```

AhnHS_MakeAckMsg

DESCRIPTION

クライアントで使⽤します。サーバーから受け取った暗号化された CRC 要求メッセージを復号化し、ゲームファイルの各種情報を得て暗号化された CRC 応答メッセージを生成します。

SYNTAX

```
int __stdcall
_AhnHS_MakeAckMsg (
    IN unsigned char *pbyReqMsg,
    OUT unsigned char *pbyAckMsg
);
```

PARAMETERS

Parameter	Type	Description
pbyReqMsg	unsigned char *	サーバーから送信した暗号化されたCRC要求メッセージでバッファのサイズはHShield.hに定義されたSIZEOF_REQMSGです。
pbyAckMsg	unsigned char *	サーバーに送信する暗号化されたCRC応答メッセージでバッファのサイズはHShield.hに定義されたSIZEOF_ACKMSGです。

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

ERROR_ANTICPNT_MAKEACKMSG_INVALIDPARAM (Value = 0x10000)

内容: 入力値が正しくありません。

原因: pbyReqMsg と pbyAckMsg が NULL です。

ERROR_ANTICPNT_MAKEACKMSG_INITCRYPT (Value = 0x10012)

内容: 暗号化/復号化の初期化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

Example

_AhnHS_MakeAckMsg 関数を呼び出したサンプルは次のとおりです。

サンプル

```
dwRet = _AhnHS_MakeAckMsg( byReqMsg,          // [in]
                           byAckMsg          // [out]
                           );

if( dwRet != ERROR_SUCCESS)
{
    // エラー処理
}
```

AhnHS_SaveFuncAddress

DESCRIPTION

保護するゲームクライアント内の関数ポインタを引数として受け取り、該当する関数の CRC を HackShield.crc 形態のファイルで保存します。メモリの完全性を保障するためにサーバーではこの関数から生成された HackShield.crc ファイルを管理する必要があります。関数の引数として関数ポインタを残す場合、最大で 32 個までの関数を保護することができますが、最初の引数となる関数の個数と後にくる関数ポインタの個数が一致する必要があります。

注意

この関数は EXE に存在する関数である必要があります。同一名の関数（オーバーローディング）についてはサポートしていません。

SYNTAX

```
int __stdcall
_AhnHS_SaveFuncAddress (
    IN unsigned int unNumberOfFunc,
    ...
);
```

PARAMETERS

Parameter	Type	Description
unNumberOfFunc	unsigned int	保護する関数の個数
...	...	可変引数である関数ポインタを入力します。

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

ERROR_ANTICPNT_SAVEFUNCADDRESS_INVALIDPARAM (Value = 0x10020)

内容: 入力値が正しくありません。

原因: 入力フォーマットが正しくありません。

対処方法:

ERROR_ANTICPCNT_SAVEFUNCADDRESS_INITCRYPT_FAIL (Value = 0x10023)

内容: 暗号化/復号化の初期化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

**ERROR_ANTICPCNT_SAVEFUNCADDRESS_DECRYPTMESSAGE_FAIL
(Value = 0x10024)**

内容: メッセージの復号に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

**ERROR_ANTICPCNT_SAVEFUNCADDRESS_ENCRYPTMESSAGE_FAIL
(Value = 0x10029)**

内容: メッセージの暗号化に失敗しました。

原因:

対処方法: アンラボにお問い合わせください。

6. モニタリングサービス機能

6.1. 概要

AhnLab HackShield Hacking Monitoring System(HSMS)は、HackShield を適用したゲームへのハッキング攻撃とエラーを集中的に監視します。ハッキング攻撃とエラーはゲームサービスへダメージを与えることができます。そして HSMS はハッキング攻撃とエラーをリアルタイムでモニタリングすることによりダメージを防ぐことができます。また、HSMS は、レポート機能を提供し、サービスの状況、統計情報を出力できます。

機能

リアルタイムハッキング/エラーモニタリング

HackShieldを適用したプログラムとデータに対するハッキングとエラー状況をリアルタイムにモニタリングすることができます。

レポート

収集したログをベースに各種統計情報を提供します。

ポリシー管理

状況に応じて各種ポリシーを設定し、データベースのバックアップを実行できます。

特徴

インターフェイス関数(API)の提供

Ehsvc の機能を容易にりようできるようにインターフェイス DLL とライブラリを提供します。提供されたインターフェイス DLL とライブラリを利用してゲームファイルの整合性のチェックや HackShield の運用状況を確認することができます。

モニタリングサーバープログラム

Oracle DBを使ってリアルタイムにハッキング・エラー情報の管理や参照ができるように、モニタリングプログラムを提供します。

テストプログラム

テストプログラム Amazon.exe は、Ehsvc で提供する API を利用して実装されています。Amazon.exe は、HackShield と Ehsvc のテスト機能を実装しています。

システムアーキテクチャ

HShield.lib、EhSvc.dllを提供してサーバーとクライアントにDLLとライブラリ形式で適用されます。モニタリングサービスのアーキテクチャと動作原理は以下のとおり:

EhSvc.dll (インターフェイス dll)

ハッキング攻撃やエラーが発生した際に情報をモニタリングサーバーに通知するための基本的なAPI を提供します。

HShield.lib (HackShieldライブラリ)

ハッキング攻撃やエラーが発生した際に情報をモニタリングサーバーに通知するための基本的なAPI を提供します。

HSMS_Setup.exe (モニタリングサーバーサイドインストールファイル)

クライアントから受信したデータに対して DBを利用して Web上で管理するためのプログラムです。

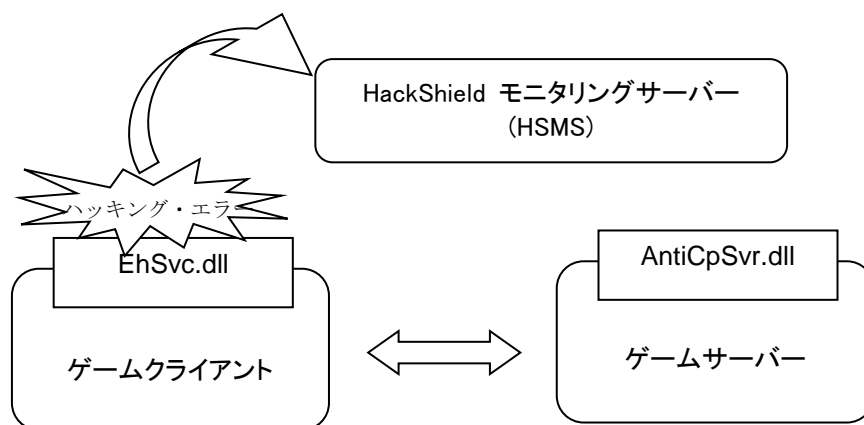


図. 6-1 モニタリングサービスの動作

ゲームクライアントが _AhnHS_StartMonitor 関数を呼び出すと、クライアントはハッキングやエラーが発生するとモニタリングサーバーに通知します。

モニタリングサーバーは、ゲームコードによりクライアントプログラムを管理します。クライアントから送信されたデータは、ゲームコードと関連付けられ DB に格納されます。格納されたデータは、Web 経由で簡単にアクセスすることが可能です。

6.2. Application Programming

Ehsvcで提供するAPI を使ってモニタリングサーバーに関する情報及びユーザー情報をセッティングします。

参考

本書で使用しているサンプルコードは Microsoft Visual C++ 6.0 を対象とした C/C++ 言語により作成されています。プログラミングに使われている言語は各プログラムの特性やシステム環境によって変更することができます。

プログラミング適用方法

プログラミングを始める前に次のような準備作業を行います。

1. Monitoring関連ファイル

Monitoring関連ファイル

表 6-1 Monitoring関連ファイル

File name	Installation folder	Description
HShield.h	[プログラムソースフォルダ]	クライアント用ヘッダーファイル
HShield.lib	[プログラムソースフォルダ]	クライアント用ライブラリファイル
EhSvc.dll	[プログラムソースフォルダ]	クライアント用 dll ファイル

6.2.1.2.適用方法

サーバー適用方法

1. HackShieldモニタリングサーバーインストールガイドを参照してモニタリングサーバーをインストールしてください。
2. HackShieldモニタリングサーバー運用ガイドを参照してモニタリングサーバーを運用してください。

クライアント適用方法

1. HShield.lib ファイルをプロジェクトに追加します。
2. HShield.h ファイルをソースに追加します。
3. AHNHS_EXT_ERRORINFO 構造体を宣言し、初期化します。

```
AHNHS_EXT_ERRORINFO HsExtError = { 0, };
```

4. HsExtError 構造体のメンバをセットします。
szServer(モニタリングサーバーアドレス)
szUserId(ユーザーアカウント)
szGameVersion(ゲームバージョン)

```
HsExtError.szServer = "127.0.0.1" //モニタリングサーバーアドレス  
HsExtError.szUserId = "Test" //ユーザーアカウント  
HsExtError.szGameVersion = "3.0.0.1" //ゲームバージョン
```

参考

モニタリングサーバーIP/Port情報は、HSUpdate.env ファイルを利用して後で、変更することができます。

5. 上記のステップで作成した構造体を引数に使用して EhSvc.dllのブルパスを送信して _AhnHS_StartMonitor関数を呼び出します。

注意

必ず `_AhnHS_StartMonitor` 関数は `_AhnHS_Initialize`関数の前に呼び出してください。
そうでない場合、HackShieldが起動する前のエラーは送信しません。

注意

HackShield アップデートライブラリと HackShield ライブラリを同一ゲームモジュールに適用する場合、必ず HackShield アップデートライブラリを適用した後、`_AhnHS_StartMonitor` 関数を呼び出す必要があります。

そうしないと、正常にアップデートできません。

```
lstrcat (szFullFileName, _T("¥¥HShield¥¥EhSvc.dll"));

dwRet = _AhnHS_StartMonitor (  HsExtError    // [in]
                               szFullFileName // [in]
                               );

if( dwRet != ERROR_SUCCESS)
{
    // 失敗が発生するとモニタリング機能だけが動作できないので
    // エラーに関するログだけを記録するようにする。
}

dwRet = _AhnHS_Initialize ( ...
```

6. 4番ステップでユーザーID情報が受信できない場合、ユーザーIDの受信ができるタイミングに `_AhnHS_SetUserId`関数を呼び出したらユーザーIDを設定できます。

6.3. Application Programming Interface

AhnHS_StartMonitor

DESCRIPTION

ハッキングとエラーが発生した時にデータを送信するサーバー情報を設定します。
_AhnHS_Initialize 関数の前に呼び出します。

SYNTAX

```
int  
_stdcall  
_Ahn_StartMonitor ( IN AHNHS_EXT_ERRORINFO HsExtErrorInfo,  
                   IN LPCSTR szFileName  
                   );
```

PARAMETERS

Parameter	Value	Description
HsExtErrorInfo	AHNHS_EXT_ERRORINFO	構造体(サーバーURLアドレス、ユーザーID、ゲームバージョン)
szFileName	LPCSTR	EhSvc.dllのプルパス

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 関数の呼び出しに成功した時に返される値です。

原因: Normal

対処方法:

HS_ERR_INVALID_FILES (Value = 0x1C001)

内容: 入力データが不正です。

原因: パラメータが NULL です。

対処方法: HsExtErrorInfo と szFileName が正しいか確認してください。

HS_ERR_UNKNOWN (Value = 0x1C002)

内容: 不明なエラー

原因: 関数の内部で例外が発生したか関数の構造的な問題の可能性があります。

対処方法: HShield.log ファイルと AhnReport を収集してアンラボに送ってご相談ください。

Example

_AhnHS_StartMonitor関数を呼び出したサンプルは次のとおりです。

サンプル

```
AHNHS_EXT_ERRORINFO HsExtError;    // HShield.hに定義された構造体

HsExtError.szServer = "127.0.0.1"    //モニタリングアドレス
HsExtError.szUserId = "Test"         //ユーザー ID
HsExtError.szGameVersion = "3.0.0.1" //ゲームバージョン

lstrcat (szFullFileName, _T("¥¥HShield¥¥EhSvc.dll" ));

dwRet = _AhnHS_StartMonitor (  HsExtError    // [in]
                               szFullFileName // [in]
                               );

if( dwRet != ERROR_SUCCESS)
{
    // 失敗が発生するとモニタリング機能だけが動作できないので
    // エラーに関するログだけを記録するようにする。
}

dwRet = _AhnHS_Initialize ( ...
```

AhnHS_SetUserId

DESCRIPTION

モニタリングサーバーに送信するメッセージのユーザーIDを保存します。
_AhnHS_StartMonitor 関数でもユーザ ID情報受信しますが、HackShieldを初期化する時ユーザー情報がなくなる場合があります。それでゲームクライアントがユーザーIDを把握したタイミングにこの関数を呼び出してユーザー情報を取得してください。ユーザーIDを取得する前にはユーザーID情報無しでエラーが送信されます。

SYNTAX

```
void _stdcall  
_AhnHS_SetUserId ( IN LPCSTR szUserID )
```

PARAMETERS

Parameter	Value	Description
szUserUD	LPCSTR	ゲームクライアントのユーザー情報

RETURN VALUE

None.

Example

_AhnHS_SetUserId関数を呼び出したサンプルは次のとおりです。

サンプル

```
_AhnHS_SetUserId ( IN LPCSTR szUserID );
```

7. LMP 機能

7.1. 概要

Local Memory Protection (LMP)は一部※パッカーを使った場合、既存のサーバー連動機能のメモリ保護方式で保護できない場合の対策でクライアント側からメモリの偽造/改ざんの検知することができる機能です。これは既存のサーバー連動のメモリ偽造/改ざん機能と似てましたがサーバーを利用しないまま、クライアント側自体的なメモリ領域の保護ができます。

※ Themidaパッカーでメモリ保護のために毎回実行される時、毎回新しいメモリアドレスのコードを変更しているため、サーバーでメモリCRCをチェックするサーバー連動方式と競合が発生する。

LMP 1.0

CSInspector ツールを使用して EXE ファイルまたは、DLL ファイルに保護を適用することができます。THEMIDA の API Wrappingオプションはサポートしていません。

保護ファイル: EXE, DLL files

利用ツール: CSInspector.exe

LMP 2.0

LMP 1.0の既存の機能は以前のままで、THEMIDAの API Wrappingをサポートするために開発されたローカルメモリ保護機能機能であり、サーバー連動のhsbファイルの生成ツールを使ってゲームクライアントファイルに適用できます。

保護ファイル: EXE,DLL files

利用ツール: HSBGen.exe

注意

LMP1.0 と 2.0 を混在し適用しないことを推奨します。

機能

実行ファイルのメモリ保護 機能(LMP 1.0、LMP 2.0サポート)

メモリにある実行ファイルのコードエリアの改ざんを検出します。

DLL ファイルのメモリ保護 (LMP 1.0、LMP 2.0サポート)

メモリにある特定の DLL ファイルのコードエリアの改ざんを検出します。直接ビルドされていない DLL の場合も保護機能は有効です。

注意

動的にローディングされるDLL を保護する場合、HackShield初期化(AhnHS_Initialize)→開始(AhnHS_StartService)関数を呼び出した以後に該当のDLLを読み込む順番なら、DLL読み込んだ後、_AhnHS_IsModuleSecure(szDllPath) 関数を呼び出す必要があります。

メモリ完全性チェック機能 (LMP 1.0、LMP 2.0サポート)

LMP機能が起動する前に、メモリが改ざんされてしまっていないか確認するため、システムファイルがフックされていなか、またメモリの整合性を確認する機能を提供します。

特徴

オプション提供

InitializeのAHNHS_CHKOPT_LOCAL_MEMORY_PROTECTIONオプションを適用したら該当の機能が実行されます。

コールバックメッセージ提供

対応する関数において、メモリ改ざんが検出された場合、HackShield のコールバック関数がAHNHS_ACTAPC_DETECT_MEM_MODIFY_FROM_LMP を送信します。

テストプログラムの提供

テストプログラム Amazon.exe は、Ehsvc で提供する API を利用して実装されています。Amazon.exe は、HackShield と Ehsvc のテスト機能を実装しています。

システムアーキテクチャ

CSInspector.exeを使って保護するモジュールにセクション情報を入力したHShield.lib、EhSvc.dllを提供してクライアントにDLLとライブラリ形式で適用されます。

EhSvc.dll (インターフェイス dll)

ハッキング攻撃やエラーが発生した際に情報をモニタリングサーバーに通知するための基本的なAPI を提供します。

HShield.lib (HackShieldライブラリ)

ハッキング攻撃やエラーが発生した際に情報をモニタリングサーバーに通知するための基本的なAPI を提供します。

CSInspector.exe (保護モジュール設定UTIL) (LMP 1.0)

HackShield がモジュールのコードエリアを保護するために、クライアントファイルまたは、サードパーティのモジュールにセクション情報を入力します。

HSBGen.exe (保護モジュール設定UTIL) (LMP 2.0)

HackShield がモジュールのコードエリアを保護するために、クライアントファイルまたは、サードパーティのモジュールにセクション情報を入力します。

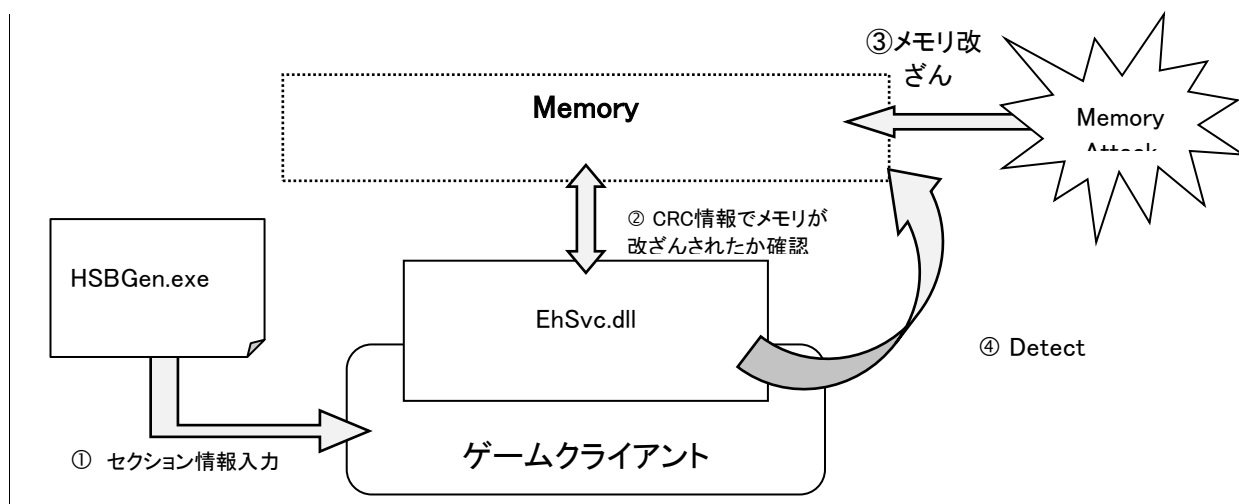


図. 7-1 Local Memory Protection

LMP機能の動作原理は以下のとおり。:

HSBGen.exe ツールを利用して、ゲームクライアントの DLL ファイルのセクションデータを特定の PE ヘッダーに保存します。

次に、HackShield 実行時に セクションデータを ゲームクライアントのファイルまたは、DLL の PE ヘッダーから読み込みます。その時、メモリ改ざんが行われていないか確認します。

セクションデータをもとに現時点のメモリ状況の CRC データを作成します。メモリ改ざん

が行われていないか、作成した CRC を基準にして定期的にチェックを行います。

7.2. Application Programming

Ehsvc で提供される HackShield 初期化関数 (_AhnHS_Initionlize)によりLMP 機能を利用することができるようになります。

参考

本書で使用しているサンプルコードは Microsoft Visual C++ 6.0 を対象とした C/C++ 言語により作成されています。プログラミングに使われている言語は各プログラムの特性やシステム環境によって変更することができます。

プログラミング適用方法

LMP機能を利用してプログラミングする前に以下のことを確認します。

7.2.1.1.LMP関連ファイル

LMP関連ファイル

表 7-1 LMP関連ファイル

パス	ファイル名	インストールフォルダ	説明
[SDK]¥Include¥	HShield.h	[プログラムソースフォルダ]	クライアント用ヘッダーファイル
[SDK]¥Lib	HSheild.lib	[プログラムソースフォルダ]	クライアント用ライブラリファイル
[SDK]¥Bin¥Win¥x86¥Util	CSInspector.exe		保護モジュール設定Utility
[SDK]¥Bin¥Win¥x86¥AntiCrack¥	HSBGen.exe		保護モジュール設定Utility

7.2.1.2.適用方法

クライアント適用方法

1. HShield.lib ファイルを作業するプロジェクトに追加します。
2. 提供されている HShield.h ファイルを作業するソースに追加します。
3. 該当のソースで_AhnHS_Initialize関数を呼び出す時、第5引数に以下のオプションを追加してください。

```
// _AhnHS_Initialize関数を呼び出す時のオプションフラッグを定義します。
// (既存のオプションに追加)
dwOption = AHNHS_CHKOPT_ALL |
AHNHS_CHKOPT_LOCAL_MEMORY_PROTECTION;

// _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 ( szFullPath,
                          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,             // オプションフラッグ
                          AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL
                          );
...

```

4. HackShield に関連するイベント転送関数を作成したところに AHNHS_ACTAPC_DETECT_MEM_MODIFY_FROM_LMP処理を追加します。

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

    // 場合によってエラーメッセージを出力します。
    switch ( lCode )
    {
        // LMP Callback
        // 参照: LMPコールバックからは 改ざんされたモジュール名とページ上のパスも
        // パラメータとして送信されますが、
        // これをユーザーに公開する必要はありません。
        case AHNHS_ACTAPC_DETECT_MEM_MODIFY_FROM_LMP:
            wsprintf(szMsg, "モジュールからメモリ改ざんが検知されました。¥n");
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;
        ....
    }
}

```

以下をパラメータとして送信します。:

lCode AHNHS_ACTAPC_DETECT_MEM_MODIFY_FROM_LMP(0x10705)

pParam

“改ざんされたモジュール名 (モジュールベースアドレス):

改ざんされた実際のページアドレス”

パラメーターで改ざんされたモジュールの名前とページアドレスが返されます。

この情報は改ざん発生時に情報として活用されゲームクライアントのユーザー

には知らせる必要ありません。

保護モジュール適用 - CSInspector.exe (LMP 1.0)

パッキングしたモジュールを配布する場合、パッカーを実行する前に CSInspector.exe を使用して下さい。

1. LMP機能を利用するため、提供のCSInspector.exe使って保護する対象モジュールに適用してください。
2. CSInspector.exe は、コマンドラインで利用します。以下は実行例です。

```
C:\> CSInspector.exe Target.exe
```

3. 追加で保護するDLLファイルがある場合、同じ方法でCSInspector.exeを適用します。

```
C:\> CSInspector.exe Target.dll
```

4. 正常に実行すると SUCCESS メッセージが表示されます。
5. CSInspector.exe を実行した後 パッカーやCRCの生成を行います。

参考

CSInspector の詳細については、[9.7 CSInspectorツール](#) を参照してください。

保護モジュール適用- HSBGen.exe (LMP 2.0)

参考

HSBGen の詳細については、[9.4 HSBGen ツール使い方](#) を参照してください。

その他の機能

AhnHS_IsModuleSecure

CSInspector.exe (LMP1.0) と HSBGen.exe (LMP2.0) によって保護されているモジュールの整合性を確認できます。

_AhnHS_IsModuleSecure関数を呼び出したサンプルは次のとおりです。

```
サンプル
_AhnHS_Initialize      (..) ;                // HackShield初期化
_AhnHS_StartService (..);                // HackShield開始

LoadLibrary("C:¥¥GAME¥¥GameEngine.dll"); // DLL-loading

// 必ずHackShieldが正常に初期化されて、保護対象のDLLが読み込まれた後に
// 呼び出します。
BOOL bRet = _AhnHS_IsModuleSecure ("C:¥¥GAME¥¥GameEngine.dll");
```

説明

もし bRet が TRUE の場合、CSInspector で入れた情報が正常に入力されています。
FALSE の場合、情報は改ざんされているかまたは、情報が存在しません。
この機能は、ある DLL をダミーの DLL に置き換えるようなハッキング攻撃を検出することができます。

_AhnHS_IsModuleSecure

DESCRIPTION

LMP を利用する場合、
CSInspector (LMP1.0) または、HSBGen.exe (LMP2.0)を利用して保護する DLL内に記録された情報の整合性を確認します。

SYNTAX

BOOL _stdcall _AhnHS_IsModuleSecure (IN LPCSTR szModulePath)

PARAMETERS

Parameter	Value	Description
szModulePath	LPCSTR	DLL path (フルパス)

RETURN VALUE

TRUE

内容: CSInspectorを利用して入力した情報が有効です。

原因: Normal

FALSE

内容: CSInspectorを利用して入力した情報が無効です。

原因: 保護するDLL が破損しているか他のものに置き換えられています。

対処方法: CSInspectorを利用して保護するDLL内に情報を正常に入力したかを再度確認してください。

Example

_AhnHS_IsModuleSecure関数を呼び出したサンプルは次のとおりです。

```
サンプル  
BOOL bRet = _AhnHS_IsModuleSecure ("C:¥¥GAME¥¥GameEngine.dll");
```

注意

– この関数はHackShieldが正常に初期化(_AhnHS_Initialize)された以後に呼び出してください。

サーバー連動利用時の注意点

CSInspector を利用して旧サーバー連動や拡張サーバー連動をLMPと同時に利用する際の注意事項は以下のとおり:

1. 旧サーバー連動

- ゲームクライアントのファイルにLMP機能を適用するには、必ずCSInspector.exeで作業した後にAntiCpSvrTool.exeを利用してHackShield.crcを作成する必要があります。
- もうサーバー連動を利用している場合、該当のモジュールにLMP機能を適用した後にAntiCpSvrTool.exeを利用してHackShield.crcを再度作成する必要があります。

2. 拡張サーバー連動（ただし、HSBGen [LMP2.0] を適用する場合該当事項無し）

- ゲームクライアントのファイルにLMP機能を適用するには、必ずCSInspector.exeで作業した後にHSBGen.exe利用してanticpx.hsbを作成する必要があります。
- もう拡張サーバー連動を利用している場合、該当のモジュールにLMP機能を適用した後にHSBGen.exe利用してanticpx.hsbを再度作成する必要があります。

サポートパッカー

LMP は現在以下のパッカーをサポートしています。

表 7-2 LMPサポートパッカー

Packer name	Version	Remarks
Themida	2.0.6.5	LMP1.0はApiWrap機能のサポート不可 LMP 2.0からサポート可能
Armadillo	V6.4.0.640	パッキングオプションによって部分的にサポート可能（※参照:Doc¥Additional¥Packer_HackShield_compatibility.pdf）

8. その他の機能

8.1. データファイル/メッセージ暗号化機能

8.1.1.概要

HsCryptoUtil はデータの暗/復号化 SDK です。最近、コンピュータの発達にともなってより強力な暗号化技術が要求されており、その結果 DES (Data Encryption Standard) より進歩した AES (Advanced Encryption Standard) が標準として使用されています。HsCryptoUtil は 128bit AES を使って、より強力にデータの暗/復号化をします。

機能

データの暗号化/復号化

ファイルやメッセージの暗/復号化ライブラリは HsCryptLib.lib (Windows 用) と libhscrypt.so (Linux 用)、HsCryptLib.h を提供しており、ゲーム開発会社では簡単にメッセージやファイルの暗/復号化処理を行うことができます。

特徴

インターフェイス関数(API)の提供

HsCryptLib の機能を使ってその実行結果の値を確認できるようインターフェースライブラリを提供します。開発者は提供されているインターフェースライブラリを使ってファイルやメッセージを簡単に暗/復号化することができます。

データ暗号化プログラムの提供

ファイルの暗号化ツールである HsCryptoUtil.exe を提供し、暗号化されたファイルを HsCryptLib の API を使って復号化します。

システムアーキテクチャ

HsCryptLib は独立した実行ファイル形態ではない SDK を使ったライブラリ形態で提供されます。HsCryptLib の全体構造と動作原理は次のとおりです。

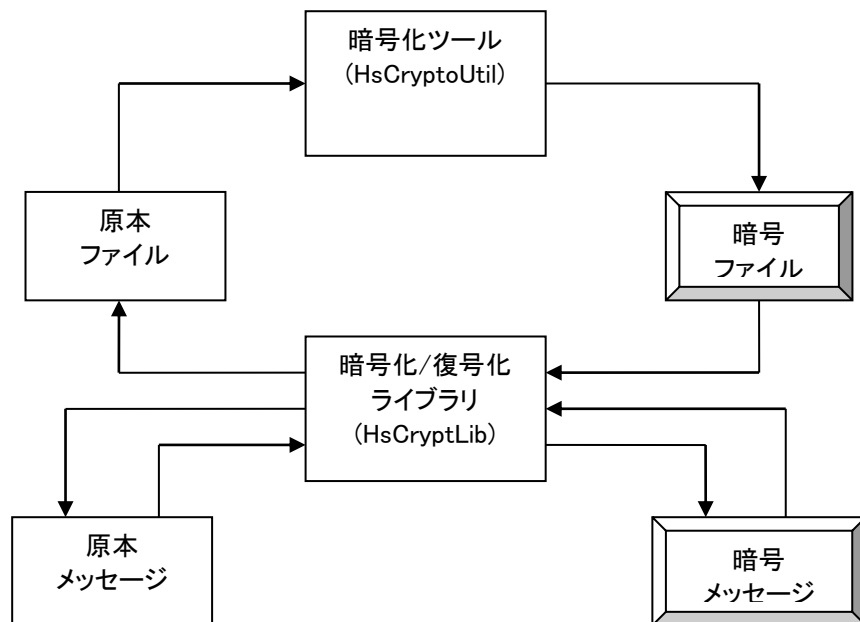


図. 8-1 HsCryptLib の全体構造と動作原理

インターフェイスライブラリ: HsCryptLib.lib(for Windows), libhscrypt.so (for Linux)

インターフェイスライブラリファイルとしてデータを暗/復号化する時に使用する API を提供します。

HsCryptoUtil プログラム

暗号化プログラムとして HsCryptLib.lib を使ってファイルを暗号化する機能を提供します。

8.1.2.Application Programming

プログラミング手順

ゲーム開発会社は次のような手順でHsCryptLib機能を使うことができます

1. 準備工程: 提供された HsCryptLib ファイルリストを確認して、必要なファイルをコピーします。
2. 暗/復号化の初期化関数の呼び出し: 暗/復号化を初期化する関数を作成してデータを暗/復号化する前に呼び出します。初期化しないとファイルおよびメッセージを暗/復号化することができません。
3. ファイルの復号化関数呼び出し: ファイルが暗号化されると、ファイルの一部あるいは全体を復号化する関数の呼び出しコードを作成してデータの特定期間や全体を復号化することができます。復号化されたデータはバッファに出力されます。
4. メッセージの暗号化関数呼び出し: メッセージを暗号化する関数の呼び出しコードを作成してメッセージを暗号化することができます。暗号化されたデータはバッファに出力されます。
5. メッセージの復号化関数呼び出し: メッセージを復号化する関数の呼び出しコードを作成してメッセージを復号化することができます。復号化されたデータはバッファに出力されます。
6. 作成されたソースコードが正常に動作するかテストします。
7. ユーザーに配布します。

プログラミング準備

HsCryptLib を使ってプログラミングを始める前に次のような準備を行います。:

HsCryptLibファイル

HsCryptLibファイル

表 8-1 HsCryptLibファイル

ファイル名	インストールフォルダ	説明
HsCryptLib.h	[プログラムソースフォルダ]	ヘッダーファイル
HsCryptLib.lib	[プログラムソースフォルダ]	Windows用ライブラリファイル :Multi thread、single threadライブラリ 提供
libhscrypt.so	[プログラムソースフォルダ]	Linux用ライブラリファイル

コンパイル設定

HsCryptLib を使った暗/復号化プログラムのプロジェクトファイルの場合 HsCryptLib.lib (libhscrypt.so) ファイルをライブラリやソースコードリストに登録する必要があります。ただし、Windows 用 HsCryptLib を使ったプロジェクトはマルチスレッド基盤ライブラリを使用するのか、シングルスレッド基盤ライブラリを使用するのか確認して、それに合った HsCryptLib.lib を利用します。

HsCrypt_InitCrypt

プログラミングの準備作業が完了したら HsCrypt_InitCrypt 関数を呼び出し、実行に成功した場合にのみ暗号化・復号化のキーの作成をすることができます。この暗号化・復号化のキーを使って暗号化・復号化することができます。

_HsCrypt_InitCrypt 関数を呼び出したサンプルは次のとおりです。

サンプル

```
typedef struct _HSCRYPT_KEYINFO
{
    BYTE  byInitKey[HSCRYPTLIB_INITKEY_SIZE]; // 初期化キー
    BYTE  AesEncKey[HSCRYPTLIB_KEY_SIZE];      //暗号キー
    BYTE  AesDecKey[HSCRYPTLIB_KEY_SIZE];      // 復号キー
} HSCRYPT_KEYINFO, *PHSCRYPT_KEYINFO;

HSCRYPT_KEYINFO HsKeyInfo;
memcpy( HsKeyInfo.byInitKey, pbyInitKey, HSCRYPTLIB_INITKEY_SIZE );

dwRet = _HsCrypt_InitCrypt ( &HsKeyInfo );
```

上記の適用サンプルで HSCRYPT_KEYINFO 構造体を宣言してから byInitKey に初期化キーの値を入力します。HsKeyInfo.byInitKey に入る初期化キーのサイズは 16byte に設定する必要があります。

_HsCrypt_InitCrypt を呼び出すと HSCRYPT_KEYINFO 構造体の AesEncKey (暗号キー) や AesDecKey (復号キー) に設定値を割り当てます。この値を使ってメッセージやファイルの暗/復号化を実行します。

注意

初期化キーによって生成された暗/復号化キーの組み合わせが一致しないとファイルやメッセージの暗/復号化をすることができません。

[_HsCrypt_GetEncMsg](#)

メッセージを暗号化する関数です。HsCrypt_InitCrypt で暗/復号化を初期化して _HsCrypt_GetEncMsg を呼び出すことでデータを暗号化することができます。この時、暗号化されたデータはバッファに出力されます。

サンプル

```
dwRet = _HsCrypt_GetEncMsg (
    byPlainMsg,           // [in] 暗号化するバッファ
    sizeof(byPlainMsg),   // [in] 暗号化するサイズ
    HsKeyInfo.AesEncKey,  // [in] 暗号化キー
    byEncMsg              // [out]暗号化されたバッファ
);
```

参考

暗号化する前のメッセージのサイズと暗号化されたメッセージのサイズは同じです。

[_HsCrypt_GetDecMsg](#)

メッセージを復号化する関数です。暗/復号化を初期化して _HsCrypt_GetDecMsg を呼び出すことでデータを復号化することができます。この時、復号化されたデータはバッファに出力されます。

サンプル

```
dwRet = _HsCrypt_GetDecMsg (
    byEncMsg,           // [in] 復号化するバッファ
    sizeof(byEncMsg),   // [in] 復号化するサイズ
    HsKeyInfo.AesDecKey, // [in] 復号化キー
    byDecMsg            // [out]復号化されたバッファ
);
```

参考

復号化する前のメッセージのサイズと復号化されたメッセージのサイズは同じです。

`_HsCrypt_FRead`

ファイルの構造体ポインタを使ってファイルの一部および全体を復号化する関数です。暗/復号化キーを初期化して `fseek` 関数を呼び出し、指定したファイルポインタに移動するとデータを復号化することができます。この時、復号化されたデータはバッファに出力されます。

サンプル

```
dwRet = _HsCrypt_FRead (
    byPlainBuf,           // [out] 復号化されたバッファ
    dwDecSize,            // [in]  復号化されるサイズ
    InputStream,          // [in]  読み込むファイルポインタ
    HsKeyInfo.AesDecKey,  // [in]  復号キー
    &dwReadLen             // [out] 復号化されたサイズ
);
```

ブロック暗号 (Block Cipher) を使って暗/復号化します。復号化する部分をブロックで分けてブロックを復号化し、必要な部分のみバッファに保存して返します。

ファイル全体を復号化するには `fseek` 関数を使ってファイルの構造体ポインタをファイルの先頭に設定してから `_HsCrypt_FRead` の 2 番目のパラメータにファイル全体のサイズを入力します。

注意

対象ファイルを暗号化した暗号キーの対の復号キーを入力する必要があります。キーの管理が難しい場合には暗/復号キーを初期化した初期化キーと `_HsCrypt_InitCrypt` を使って生成された復号キーを使用します。

8.1.3.Application Programming Interface

_HsCrypt_InitCrypt

DESCRIPTION

暗/復号化関数を初期化します。

SYNTAX

```
DWORD __stdcall
_HsCrypt_InitCrypt (
    IN OUT PHSCRYPT_KEYINFO pHsKeyInfo
);
```

PARAMETERS

Parameter	Description
PHSCRYPT_KEYINFO	[in][out] 暗/復号 キー構造体
Structure Definition	
typedef struct_HSCRYPT_KEYINFO	
{	
BYTE byInitKey[HSCRYPTLIB_INITKEY_SIZE];	初期化キー(16bytes)
BYTE AesEncKey[HSCRYPTLIB_KEY_SIZE];	暗号キー(550bytes)
BYTE AesDecKey[HSCRYPTLIB_KEY_SIZE];	復号キー(550bytes)
} HSCRYPT_KEYINFO, *PHSCRYPT_KEYINFO;	

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 初期化に成功した時に返される値です。

ERROR_HSCRYPTLIB_INITCRYPT_INVALIDPARAM (Value = 0x0001B002)

内容: 不正なパラメータが入力された時に返される値です。

原因: 渡されたポインタは NULL です。

対処方法: 引数の内容が正しいか確認します。

Others

WIN32 Defined Error (Value = WIN32 Defined)

REMARKS

データを暗/復号化するために呼び出す関数として、暗/復号化に必要ないくつかのデータ値を設定します。HSCRYPT_KEYINFO 構造体の byInitKey を割り当ててから _HsCrypt_InitCrypt 関数を呼び出すと AesEncKey (暗号キー) や AesDecKey (復号キー) を求めることができます。このキーを使ってメッセージやファイルの暗/復号化を行います。

_HsCrypt_GetEncMsg

DESCRIPTION

メッセージを暗号化して暗号化されたデータをバッファに出力します。

SYNTAX

```
DWORD __stdcall  
_HsCrypt_GetEncMsg (  
    IN PBYTE pbyInput,  
    IN UINT nInLength,  
    IN PBYTE pAesEncKey,  
    OUT PBYTE pbyOutput  
);
```

PARAMETERS

Parameter	Description
pbyInput	[in] 暗号化するバッファ
nInLength	[in] 暗号化するサイズ
pAesEncKey	[in] 暗号化キー
pbyOutput	[out] 暗号化されたバッファ

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: メッセージの暗号化に成功した時に返される値です。

原因: None

対処方法: None

ERROR_HSCRYPTLIB_GETENCMMSG_INVALIDPARAM (Value = 0x0001B003)

内容: 不正なパラメータが入力された時に返される値です。

原因: 引数の内容が正しくありません。

対処方法: 引数が NULL または、'0' でないか確認します。

その他

WIN32 Defined Error (Value = WIN32 Defined)

_HsCrypt_GetDecMsg

DESCRIPTION

メッセージを復号化して復号化されたデータをバッファに出力します。

SYNTAX

```
DWORD __stdcall  
_HsCrypt_GetDecMsg (  
    IN PBYTE pbyInput,  
    IN UINT nInLength,  
    IN PBYTE pAesDecKey,  
    OUT PBYTE pbyOutput  
);
```

PARAMETERS

Parameter	Description
pbyInput	[in] 復号化するバッファ
nInLength	[in] 復号化するサイズ
pAesEncKey	[in] 復号化キー
pbyOutput	[out] 復号化されたバッファ

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: メッセージの暗号化に成功した時に返される値です。

原因: None

対処方法: None

ERROR_HSCRYPTLIB_GETDECMMSG_INVALIDPARAM(Value = 0x0001B004)

内容: 不正なパラメータが入力された時に返される値です。

原因: 引数の内容が正しくありません。

対処方法: 引数が NULL または、'0' でないか確認します。

その他

WIN32 Defined Error (Value = WIN32 Defined)

_HsCrypt_FRead

DESCRIPTION

ファイルの指定した部分のみを復号化してデータをバッファに出力します。

SYNTAX

```
DWORD __stdcall  
_HsCrypt_FRead (  
    OUT LPVOID lpOutBuffer,  
    IN DWORD dwDecryptSize,  
    IN FILE *pInputStream,  
    IN PBYTE pAesDecKey,  
    OUT PDWORD pdwReadLen  
);
```

PARAMETERS

Parameter	Description
lpOutBuffer	[out] 復号化されたバッファ
dwDecryptSize	[in] 復号化するサイズ
pInputStream	[in] 復号化する構造体ポインタ
pAesDecKey	[in] 復号化キー
pdwReadLen	[out] 復号化されたサイズ

RETURN VALUE

ERROR_SUCCESS (Value = 0x00000000)

内容: 復号化に成功した時に返される値です。

原因: None

対処方法: None

ERROR_HSCRYPTLIB_FREAD_INVALIDPARAM (Value = 0x0001B005)

内容: 正しくないパラメータを入力した時に返される値です。

原因: 引数の内容が正しくありません。

対処方法: 引数が NULL または、'0' でないか確認します。

ERROR_HSCRYPTLIB_FREAD_GETFILELEN (Value = 0x0001B009)

内容: ファイルサイズの取得に失敗した時に返される値です。

原因: ファイルハンドルは正常ではありません。

対処方法: ファイルハンドルを確認します。

ERROR_HSCRYPTLIB_FREAD_SIZEZERO (Value = 0x0001B00B)

内容: ファイルサイズが 0 の時に返される値です。

原因: 復号するファイルがありません。

対処方法: ファイルに異常がないか確認します。

ERROR_HSCRYPTLIB_FREAD_GETPOSITION (Value = 0x0001B00A)

内容: ファイルポインタの現在地取得に失敗した時に返される値です。

原因: ファイルハンドルが正しくありません。

対処方法: ファイルハンドルを確認します。

ERROR_HSCRYPTLIB_FREAD_FSEEK (Value = 0x0001B00C)

内容: 現在のファイルポインタ位置のブロックへ移動するのに失敗した時に返される値です。

原因: ファイルハンドルが正しくありません。

対処方法: ファイルハンドルを確認します。

ERROR_HSCRYPTLIB_FREAD_DECRYPT_RANGE (Value = 0x0001B006)

内容: 復号化するブロックのサイズがファイルより大きい時に返される値です。

原因: ファイルよりも復号しようとしているブロックの方が大きい。

対処方法: dwDecryptSizeの値がファイルサイズより大きいかを確認します。

ERROR_HSCRYPTLIB_FREAD_DECRYPT_FREAD (Value = 0x0001B007)

内容: ファイルの読み取りに失敗した時に返される値です。

原因: ファイルハンドルが正常でないか、ファイルがロックされています。

対処方法: ファイルハンドルや他のプロセスが該当のファイルにアクセスしているかを確認します。

ERROR_HSCRYPTLIB_FREAD_DECRYPT_GETDECMSG (Value = 0x0001B008)

内容: メッセージの復号化に失敗した時に返される値です。

原因: API内部エラーです。

対処方法: アンラボにお問い合わせください。

ERROR_HSCRYPTLIB_EXCEPTION (Value = 0x0001B001)

内容: 例外が発生した時に返される値です。

原因:

対処方法: アンラボにお問い合わせください。

その他

WIN32 Defined Error (Value = WIN32 Defined)

8.2. ユーザー権限実行サポート機能

8.2.1.概要

HsUserUtil は NT 系の OS を使用している場合、管理者アカウントではない一般ユーザーアカウントでログオンした状態で HackShield のゲームハッキング防御機能が正常に動作するように設定する機能です。

機能

非管理者アカウントのゲームサポート

HsUserUtil.lib を提供してゲームクライアントや HackShield に管理者アカウントではなく一般ユーザーアカウントでログオンしてもゲームを実行でき、HackShield のゲームハッキング防御機能が正常に動作するようサポートしています。

特徴

インターフェイス関数(API)の提供

HsUserUtil の機能を使ってその実行結果の値を確認できるようインターフェイスライブラリを提供します。開発者は提供されているインターフェイスライブラリを使い、固有ポリシーに従ってゲームクライアントや HackShield を一般ユーザーアカウントで実行するようにプログラミングすることができます。

テストプログラムの提供

HsUserUtil の API を使って実行したテスト用ゲームクライアントのプログラムである HsUserUtilTest.exe を提供します。テストプログラムを参考にして HsUserUtil の機能やサンプルのコードを確認し、クライアントプログラム開発に適用します。

システムアーキテクチャ

HsUserUtil は独立した実行ファイル形態ではない SDK を使ったライブラリ形態で提供されます。HsUserUtil の全体構造と動作原理は次のとおりです。:

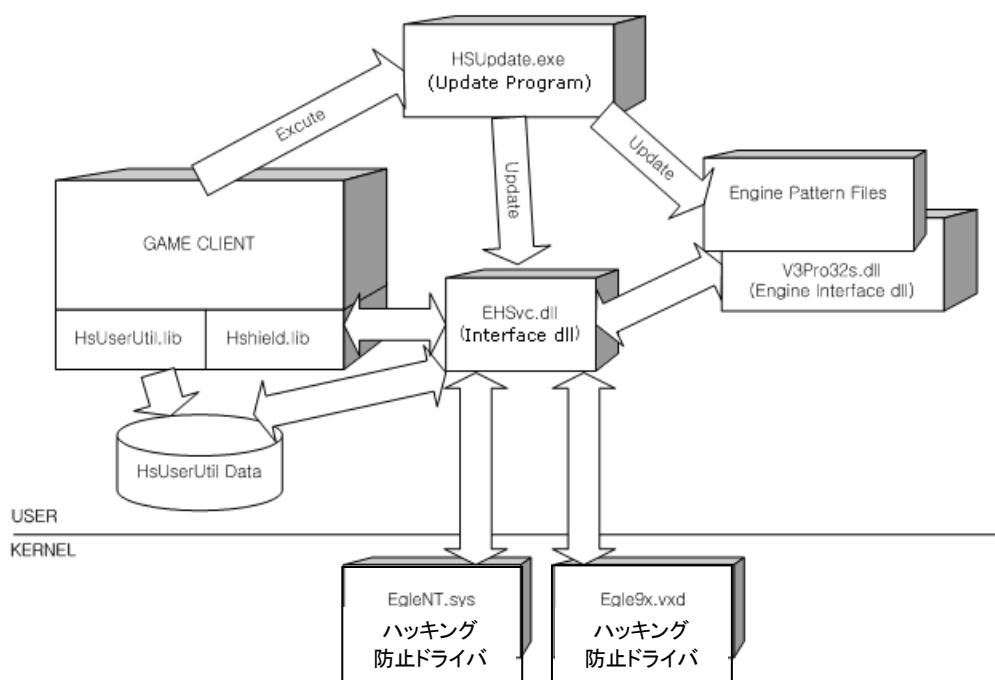


図. 8-2 HsUserUtilの全体構造と動作原理

HsUserUtil.lib (インターフェイスライブラリ)

インターフェースファイルとしてゲームプログラムが一般ユーザーアカウントで動作できるようにする API を提供します。

HsUserUtil Data

ゲームプログラムが一般ユーザーアカウントで動作できるようにするシャドウアカウント情報や関連データを保存します。

EhSvc.dll (HackShieldサービスモジュール)

HackShield サービスモジュールとしてゲームプログラムにロードされ、ハッキング防御機能を実行します。一般ユーザー権限でゲームが実行される場合、HsUserUtil データの情報を使ってゲームを正常に実行し、ゲーム防御機能が正常に動作するようサポートします。

8.2.2.Application Programming

プログラミング手順

次のような手順で HsUserUtil 機能を適用します。

参考

本書ではゲームクライアントのプログラムに HsUserUtil を使用している場合を例に説明しています。ゲームクライアントのプログラム以外にゲームランチャー (launcher) プログラムが別途存在する場合には HsUserUtil ライブラリをゲームランチャープログラムで使用することもできます。ゲームの構造に合わせて適用する必要があります。

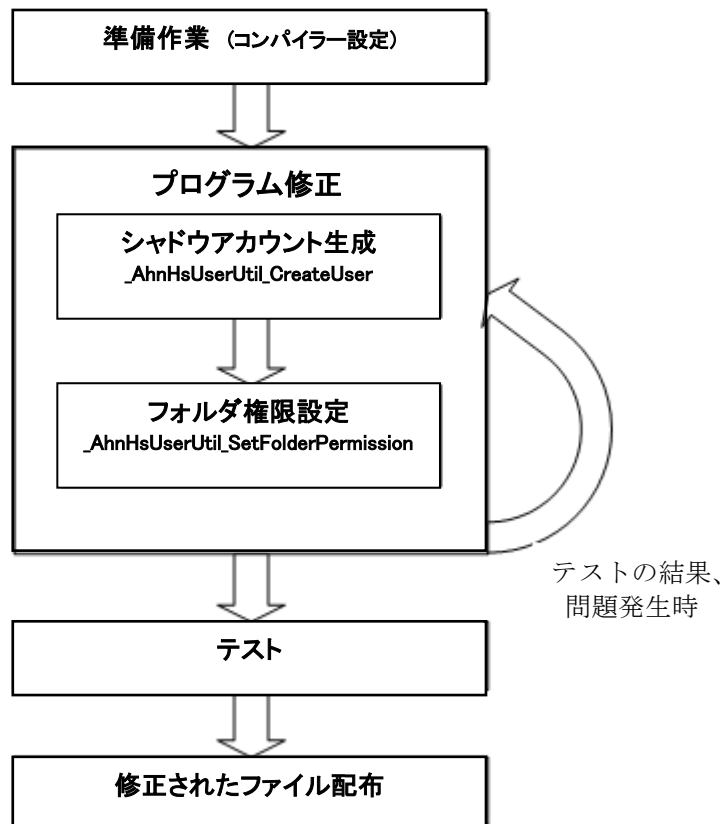


図. 8-3 HsUserUtilプログラミング手順

1. 準備工程: 提供された HsUserUtil ファイルリストを確認して、必要なファイルをコピーします。

- 2. シャドウアカウントの生成関数呼び出し: シャドウアカウントを生成する関数呼び出しコードを作成し、一般ユーザーアカウントでゲームの実行およびゲームハッキング防御機能が動作するようにします。
- 3. NTFS 権限設定関数呼び出し: サービス開始関数を呼び出した後、フォルダの NTFS 権限設定関数呼び出しコードを作成して一般ユーザーアカウントでもゲームに必要なファイルに対して書き込み機能が動作できるようにします。
- 4. ソースコードが正常に動作するかをテストします。
- 5. クライアントユーザーに配布します。

プログラミング準備

HsUserUtil を使ってプログラミングを始める前に次のような準備作業を行います。

8.2.2.1.HsUserUtilファイル

HsUserUtilファイル

表 8-2 HsUserUtilファイル

ファイル名	インストールフォルダ	説明
HsUserUtil.h	[ゲームソースフォルダ]	ヘッダーファイル
HsUserUtil.lib	[ゲームソースフォルダ]	ライブラリファイル

コンパイラ設定

HsUserUtil を使用しているゲームクライアントのプログラムのプロジェクトファイルである場合 HsUserUtil.lib ファイルをライブラリやソースコードリストに登録する必要があります。

AhnHsUserUtil_CreateUser

プログラミングの準備作業が完了したら AhnHsUserUtil_CreateUser を呼び出します。AhnHsUserUtil_CreateUser 関数の呼び出しに成功したら、一般ユーザーアカウントでもゲームを実行し、ゲームハッキング防御機能を実行することができます。

この時、_AhnHsUserUtil_CreateUser 関数は管理者アカウントでゲームプログラムまたはゲームランチャープログラムから呼び出します。

AhnHsUserUtil_CreateUser 関数を呼び出したサンプルは次のとおりです

サンプル

```
dwRet = _AhnHsUserUtil_CreateUser ( );
```

参考

_AhnHsUserUtil_CreateUser は、以前生成されたシャドウアカウントの ユーザー情報がない場合や既に生成されたシャドウアカウントのユーザー情報でログインしていない場合にのみ新しいユーザーを生成します。また、シャドウアカウント命名規則に該当するアカウントを削除するロジックが含まれているため、使用しないアカウントや不要なアカウントは新しいシャドウアカウント生成時に自動的に削除します。

[_AhnHsUserUtil_SetFolderPermission](#)

ゲームプログラムがNTFSボリュームにインストールされた場合、該当のフォルダに対して一般ユーザーアカウントには書き込み権限がなくてゲームが正常に実行できない可能性があります。例えば、ゲームモジュールに対するアップデートプログラムが起動されてもゲームがインストールされたフォルダに最新ファイルの入れ替えができなかったり、ゲーム中にゲームのデータ記録に失敗する可能性があります。

一般ユーザーアカウントに対する NTFS 書き込み権限を付与するためには _AhnHsUserUtil_SetFolderPermission 関数を使用することで一般ユーザーアカウントに対して NTFS 書き込み権限を付与することができます。この関数は次のサンプルのように呼び出します。

サンプル

```
dwRet = _AhnHsUserUtil_SetFolderPermission (“ゲームがインストールされたパス”);
```

一般ユーザーアカウントに対する NTFS 書き込み権限を付与するには該当するパスを絶対パス (Full Path) で与える必要があります。相対パスや正しくないパスを与えると誤作動することがあります。

ゲームのインストール先が NTFS ボリュームではないか NTFS ファイルのシステムを使用しない Windows 95、98、ME 系 OS の PC ではこの関数が呼び出されても何の動作もしません。

この関数を実行すると該当するパスについて Users グループに NTFS 書き込み権限を付与することになります。これを反映するには管理者権限で実行した状態でこの関数

を呼び出す必要があります。

注意

ゲーム開発者は以下のことに注意する必要があります。
ゲームクライアントの実行パスはユーザーによって異なる可能性があります。また、そのファイル権限も異なる可能性があります。予期していないフォルダにゲームがインストールされた場合、そのフォルダのNTFS権限を変更することはセキュリティの脆弱性になる可能性があります。(C:\ や デスクトップ、Windows フォルダなど)

[_AhnHsUserUtil_DeleteUser](#)

_AhnHsUserUtil_CreateUser関数で生成したシャドウアカウントを削除します。

サンプル

```
dwRet = _AhnHsUserUtil_DeleteUser ();
```

[_AhnHsUserUtil_IsEnableHSAdminRights](#)

ログインしているアカウントが HackShield を起動する権限があるか確認します。

サンプル

```
dwRet = _AhnHsUserUtil_IsEnableHSAdminRights ();
```

[_AhnHsUserUtil_CheckHSShadowAccount](#)

_AhnHsUserUtil_CreateUser 関数を利用して生成したシャドウアカウントが正しく作成されたか確認します。

サンプル

```
DWORD dwRet = HSUSERUTIL_ERR_OK;

dwRet = _AhnHsUserUtil_CheckHSShadowAccount();

switch ( dwRet )
{
    case HSUSERUTIL_ERR_NOT_NT:
        AfxMessageBox ( "HSUSERUTIL_ERR_NOT_NT" );
}
```



```

        break;
    case HSUSERUTIL_ERR_OK:
        AfxMessageBox ( "HSUSERUTIL_ERR_OK" );
        break;
    case HSUSERUTIL_ERR_SHADOWACNT_NOT_EXIST:
        AfxMessageBox ( "HSUSERUTIL_ERR_SHADOWACNT_NOT_EXIST" );
        break;
    default:
        AfxMessageBox ( "HSUSERUTIL_ERR_UNKNOWN" );
        break;
}

```

この関数は、シャドウアカウントが作成されたか確認します。

[_AhnHSUserUtil_IsAdmin](#)

ログインしているアカウントに管理者検眼があるか確認します。

サンプル

```

if ( TRUE == _AhnHSUserUtil_IsAdmin() )
{
    AfxMessageBox ( "TRUE" );
}
else
{
    AfxMessageBox ( "FALSE" );
}

```

8.2.3.Application Programming Interface

_AhnHsUserUtil_CreateUser

DESCRIPTION

一般ユーザー権限でログオンした時にゲームハッキング防御機能に使用するシャドウアカウントを生成します。

SYNTAX

```
DWORD __stdcall  
_AhnHsUserUtil_CreateUser ( );
```

PARAMETERS

None.

RETURN VALUE

HSUSERUTIL_ERR_OK (Value = 0x00000000)

内容: シャドウアカウントの生成に成功した時に返される値です。

HSUSERUTIL_ERR_NOT_ADMIN (Value = 0x0005A002)

内容: 管理者権限がありません。

原因: 現在ログオンしているアカウントが管理者アカウントではない場合に発生します。

対処方法: 実装方法にもよりますが、一般ユーザー権限で `_AhnHsUserUtil_CreateUser` を呼び出せるように適用した場合、このエラーは処理しないようにする必要があります。

HSUSERUTIL_ERR_NOT_NT (Value = 0x0005A003)

内容: NT OS ではありません。

原因: NT 系 OS のシステムではない場合に返される値です。

対処方法: OSを確認してください。

HSUSERUTIL_ERR_DELSHADOWACNT_FAIL (Value = 0x0005A005)

内容: 既存のシャドウアカウントの削除に失敗した場合に返される値です。

原因: 内部エラーです。

対処方法: アンラボにお問い合わせください。

HSUSERUTIL_ERR_DELHIDEIDINFO_FAIL (Value = 0x0005A006)

内容: Windows XP スタート画面でシャドウアカウントを隠すための情報の削除に失敗した時に返される値です。

原因: 内部エラーです。

対処方法: アンラボにお問い合わせください。

HSUSERUTIL_ERR_DELSHADOWACNTINFO_FAIL (Value = 0x0005A007)

内容: シャドウアカウント情報の削除に失敗した時に返される値です。

原因: 内部エラーです。

対処方法: アンラボにお問い合わせください。

HSUSERUTIL_ERR_ADDSHADOWACNT_FAIL (Value = 0x0005A008)

内容: シャドウアカウントが作成されなかった場合に返される値です。

原因: 内部エラーです。

対処方法: アンラボにお問い合わせください。

REMARKS

`_AhnHsUserUtil_CreateUser` 関数は Administrator 権限をもつアカウントでログオンした状態でシャドウアカウントを生成します。一般ユーザー権限で HackShield のゲーム防御機能を正常に使用するには、最初にこの関数を一度呼び出してシャドウアカウントを生成する必要があります。

AhnHsUserUtil_SetFolderPermission

DESCRIPTION

ゲームクライアントがインストールされたディレクトリに対して一般ユーザーアカウントに NTFS 書き込み権限を付与することで、一般ユーザーアカウントでログオンしてもゲームクライアントがアップデートや実行に必要なファイルの書き込み作業などを行うことができます。

SYNTAX

```
DWORD __stdcall  
_AhnHsUserUtil_SetFolderPermission (  
    LPTSTR szPath  
);
```

PARAMETERS

Parameter	Value	Description
szPath		NTFS権限を設定するところのフルパス サンプル: C:¥Program Files¥My Company¥My Game

RETURN VALUE

HSUSERUTIL_ERR_OK (Value = 0x00000000)

内容: 該当するフォルダに対して NTFS 権限の付与に成功した時に返される値です。

HSUSERUTIL_ERR_NOT_ADMIN (Value = 0x0005A002)

内容: 管理者権限がありません。

原因: 現在ログオンしているアカウントが管理者アカウントではない場合に発生します。

対処方法: 実装方法にもよりますが、一般ユーザー権限で _AhnHsUserUtil_CreateUser を呼び出せるように適用した場合、このエラーは処理しないようにする必要があります。

HSUSERUTIL_ERR_GETVOLUMEINFO_FAIL (Value = 0005A111)

内容: ドライブボリュームインフォメーションの読み取りができません。

原因: ドライブボリュームインフォメーションの読み取りの過程で発生します。

対処方法: アンラボにお問い合わせください。

REMARKS

ゲームクライアントのプログラムのインストール先がユーザーの任意で指定が可能な場合、この関数を実行する時に注意が必要です。ユーザーが C ドライブや D ドライブのようなルートディレクトリにゲームをインストールした場合、この関数が呼び出されると該当するドライブ全体に対して NTFS 権限が付与されます。これはセキュリティの脆弱性となる可能性があるため、注意して呼び出してください。

指定されたフォルダの下位に多くの下位フォルダやファイルが存在している場合にこの関数が呼び出されると NTFS 権限を設定するのに数秒から数分かかることがあります。これは最初に NTFS 権限を付与する過程でのみ発生し、権限が付与された後には影響ありません。

[_AhnHsUserUtil_DeleteUser](#)

DESCRIPTION

_AhnHsUserUtil_CreateUser. 関数で作成したシャドウアカウントを削除します。.

SYNTAX

```
DWORD __stdcall _AhnHsUserUtil_DeleteUser (void);
```

PARAMETERS

Parameter	Value	Description
void		

RETURN VALUE

HSUSERUTIL_ERR_OK (Value = 0x00000000)

内容: シャドウアカウントの削除に成功した場合に返される値です。

.

HSUSERUTIL_ERR_LOADDLL_FAIL (Value = 0x0005A004)

内容: 実行に必要な DLL の読み込みに失敗しました。

原因: システムフォルダにNETAPI32.DLL または ADVAPI32.DLL が存在しないか損傷しています。

対処方法: DLL がシステムフォルダに存在するか確認してください。

HSUSERUTIL_ERR_NOT_NT (Value = 0x0005A003)

内容: OS が NT 系ではありません。

原因: システムの OS が NT シリーズではありません。

対処方法: OS を確認してください。

REMARKS

_AhnHsUserUtil_DeleteUse 関数は_AhnHsUserUtil_CreateUser 関数で作成したシャドウアカウントを削除します。

_AhnHsUserUtil_IsEnableHSAdminRights

DESCRIPTION

ログインアカウントが HackShield を起動する権限があるか確認します。

(Admin権限がある場合やユーザー権限で、HackShieldのシャドウアカウントが生成されている場合に“Success” を返します。)

SYNTAX

DWORD __stdcall _AhnHsUserUtil_IsEnableHSAdminRights(void);

PARAMETERS

Parameter	Value	Description
void		

RETURN VALUE

HSUSERUTIL_ERR_OK (Value = 0x00000000)

内容: シャドウアカウントが正しく設定されている場合に返される値です。

HSUSERUTIL_ERR_NOT_NT (Value = 0x0005A003)

内容: OS が NT 系ではありません。.

原因: システムの OS が NT シリーズではありません。

対処方法: OS を確認してください。

REMARKS

ログインアカウントが HackShield を起動する権限があるか確認します。アカウントが管理者権限を持っているか、HackShield を起動するためのシャドウアカウントが作成されていれば、“Success” を返します。それ以外は、エラーを返します。

単 純 に HackShield の シャドウ アカウント が 存 在 す る か 確 認 す る た め に は `_AhnHsUserUtil_CheckHSShadowAccount` 関数を使用してください。

`_AhnHsUserUtil_CheckHSShadowAccount`

DESCRIPTION

`_AhnHsUserUtil_CreateUser` 関数で作成したシャドウアカウントが正しく設定されたか確認します。

SYNTAX

DWORD `_stdcall _AhnHsUserUtil_CheckHSShadowAccount();`

PARAMETERS

Parameter	Value	Description
void		

RETURN VALUE

HSUSERUTIL_ERR_OK (Value = 0x00000000)

内容: シャドウアカウントが正しく設定されている場合に返される値です。

HSUSERUTIL_ERR_NOT_NT (Value = 0x0005A003)

内容: OS が NT 系ではありません。

原因: システムの OS が NT シリーズではありません。

対処方法: OS を確認してください。

HSUSERUTIL_ERR_SHADOWACNT_NOT_EXIST (Value = 0x0005A009)

内容: HackShield シャドウアカウントが存在しません。

原因: HackShield シャドウアカウントが作成されていません。

対処方法: `_AhnHsUserUtil_CreateUser`関数を使用して HackShield シャドウアカウントが正常に作成されたか確認してください。

REMARKS

`_AhnHsUserUtil_CheckHSShadowAccount` 関数は `_AhnHsUserUtil_CreateUser`関数で作成したアカウントが正しく設定されたか確認します。

[_AhnHSUserUtil_IsAdmin](#)

DESCRIPTION

ログインアカウントが管理者権限を持っているか確認します。

SYNTAX

```
BOOL __stdcall _AhnHSUserUtil_IsAdmin ();
```

PARAMETERS

Parameter	Value	Description
void		

RETURN VALUE

TRUE

NT 系ではないか、NTシリーズの場合管理者権限を持っている。

FALSE

NT 系では管理者権限を持っていない。

REMARKS

9. ツールの使い方

9.1. AntiCpSvr ツール

機能

サーバーでCRC 情報ファイル(HackShield.crc)し、ゲームファイルやメモリやHackShieldの整合性をチェックすることができます。また、ユーザはサーバを再起動せずに新しいバージョンまたはオプションを設定するパッチを適用して、以前のバージョンに対する接続を制御できます。

作成されるたびに CRC ファイルは、他のCRC ファイルとは異なったものになります。複数のゲームサーバーがある場合はより強力なセキュリティのために、各サーバーごとに異なったCRCを適用してください。しかし同じクライアントを使う場合は同じメモリデータを有する必要があります。

9.2. AntiCpSvr ツールの使い方

UI 手動設定基盤のCRC情報ファイル生成

1. CRC File Path を入力します。[...] をクリックして HackShield.crc ファイルを生成するパスを選択します。選択したファイルのパスが表示されます。

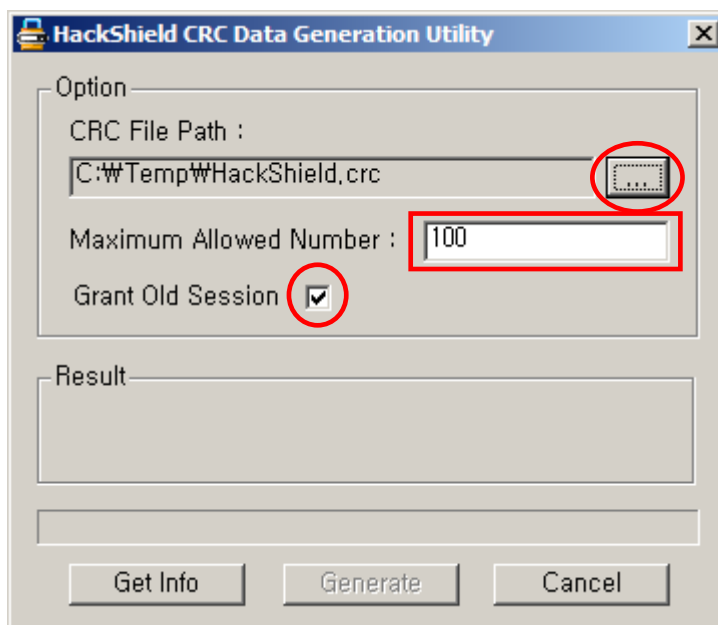


図. 9-1 HackShield CRC 情報生成ツール

2. Maximum Allowed Number を入力します。新たに生成された CRC ファイルをサーバーに適用した時、クライアントからサーバーへのアクセスに関して最大でいくつ前のバージョンまで許可するかを入力します。例えばこの値が 1 であれば、最新バージョンのみアクセスを許し、2 であれば最新のバージョンと 1 つ前のバージョンのクライアントからのアクセスを許可します。
3. Grant Old Session のオン/オフを決定します。新たに生成された CRC ファイルをサーバーに適用した時、Grant Old Session のチェック ボックスをオンにするとアクセスを許可し、オフにするとこれまでアクセスしていたクライアントのうち Maximum Allowed Number で設定した範囲を超えたバージョンを使用しているクライアントはアクセスが切断されます。
4. GetInfo をクリックして情報を入手する準備をします。
5. 新たにパッチを適用するゲームを実行させてゲームクライアント内で AhnHS_SaveFuncAddress 関数が呼び出されるまで実行し、[Generate] ボタンがアクティブになるまでゲームを進めます。

6. [Generate] ボタンがアクティブになったら実行中のゲームを終了します。ゲームが完全に終了したら 2～3 秒待機し、[Generate] をクリックして HackShield.crc ファイルを生成します。この時、新たにパッチを適用するゲームの情報を HackShield モジュールである Ehsvc.dll に保存するため、ゲームを終了してから [Generate] をクリックする必要があります。

HackShield.crc ファイルが生成されたら CRC ファイルをサーバーに適用し、CRC ファイルの生成に使うゲームクライアントファイルをパッチします。

CRC 情報ファイルの自動生成

AntiCpSvrTool は CRC ファイル生成に必要な引数情報を指定することで CRC ファイルを自動的に生成します。使用方法は下記のとおりです。

使用方法:

AntiCpSvrTool.exe CRC 情報を抽出するゲームのパス、生成される CRC ファイルのパス、Grant Old Session (1=true 0=false)、Maximum Allowed Number

注意

引数の情報は、',' (カンマ)で区切ります。

例)

Game.exe の CRC 情報を抽出して Grant Old Session = true / Maximum Allowed Number = 5 に指定し、HackShield.crc ファイルを生成するには下記のとおり実行します。

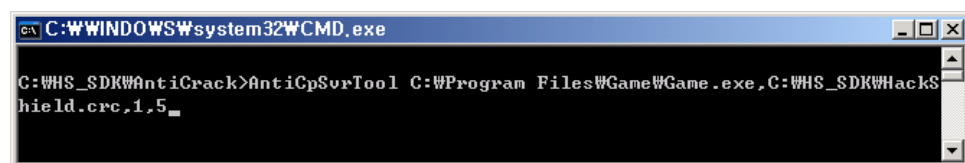


図. 9-2 Command-line タイプの AntiCpSvrTool.exe

9.3. HSBGen ツール(HackShield 専用、4.2 以降のバージョン)

機能

サーバーで HackShield Briefcase ファイル (AntiCpX.hsb) を生成してゲームファイルやメモリ、HackShield の完全性の有無を判断することができます。また、サーバーを再起動せずに新しいバージョンのパッチを適用したり、オプションを設定して以前のバージョンへの連結コントロールを可能にしたりします。

生成された HSB 情報ファイルは生成する度に異なるため、ゲームサーバーが複数であればサーバーごとに HSB 情報ファイルを適用し、セキュリティを維持します。なお、ゲームクライアントが同一である場合はメモリ情報のデータはすべて同じである必要があります。

より強力なLMP機能を使うためには配布されるゲームクライアントファイルに必要な関連情報を追加します。以前のCSInspector.exeツールから提供したところの機能をHSBGenツールから提供しています。

9.4. HSBGen ツールの使い方

UI 手動設定基盤のHSB情報ファイル生成

¥Bin¥AntiCrack¥HSBGen.exe を使って AntiCpX.hsb ファイルを生成します。

注意

クライアント実行ファイルはパッキングされる以前の原本リリースファイルである必要があります。

注意

電子署名を付けてクライアントの実行ファイルを配布する場合、HSBGen ツールを実行してから署名してください。

HSBGen.exe で AntiCpX.hsb ファイルを生成する方法は以下のとおりです。

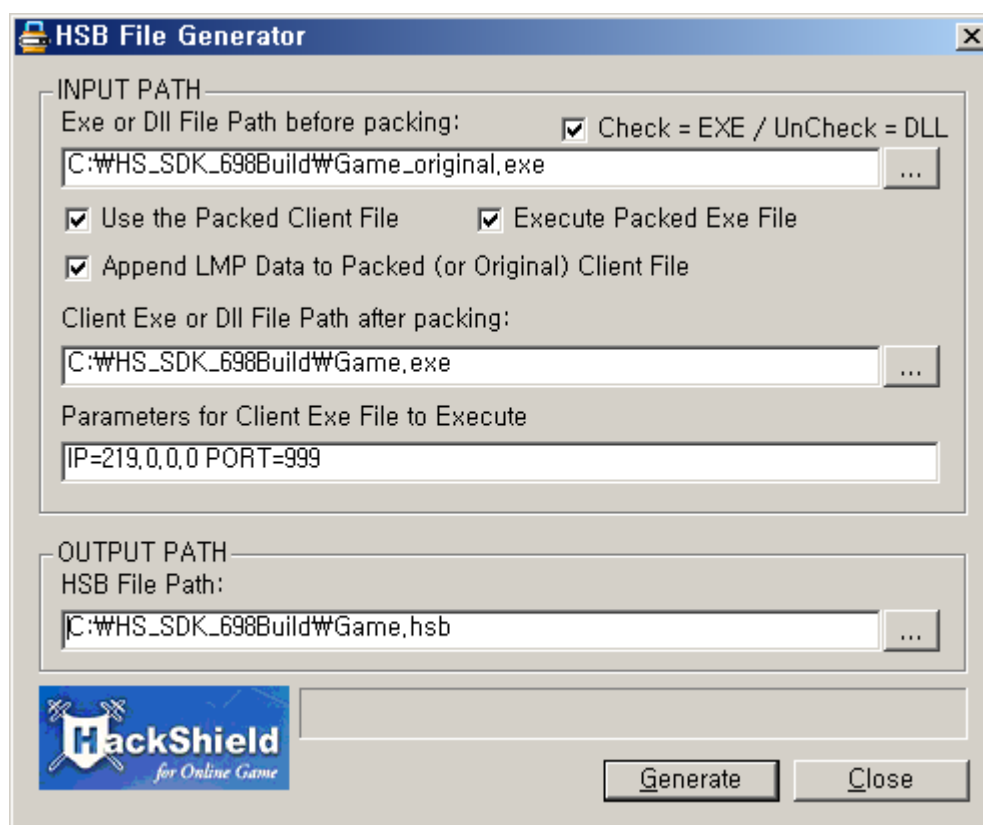


図. 9-3 HSB File Generator

1. 対象が EXE ファイルの場合、チェックします。DLL ファイルの場合チェックをはずします。

注意.

: 対象がDLLファイルの場合、対象ファイル内にLMP情報が自動的に追加されます。

2. アンラボから別途受け取った HSBGen.ini ファイルがあれば HSBGen.exe と同じフォルダへ保存します。(ゲーム別に最適化し、拡張サーバー連動の設定値を別々に適用することができます)
3. 「Exe or Dll File Path before packing」入力ボックスにパッキングされていない原本の実行ファイルパスを設定します。[...] ボタンをクリックしてファイルを選択します。
4. クライアント実行ファイルをパッキングして配布するつもりであれば、先に説明した 1 つ目のフィールドへパッキングする前のファイルパスを入力します。
5. 「Use the Packed Client File」チェック ボックスをオンにして Client Executable File Path after packing 入力ボックスへパッキングされたファイルパスを入力します。
6. 「Execute Packed Client File」は「Client Executable File Path after packing」に入力した実行ファイルを自動で起動した後、HSB ファイルを作成するかどうかを決定します。

注意

Executable Packed Client File オプションを利用する場合、

**Hsbファイル生成ボタンをクリックしたらゲームが起動するように
パラメータ情報やゲーム実行時に必要なファイルを予め準備して、
実際ゲームの実行が可能な環境を準備する必要があります。**

7. 「Parameters for Client Executable File to Execute」は「Execute Packed Client File」に入力された実行ファイルを実行する時に必要な引数などがある場合、ここにその内容を入力してください。
8. 「Append LMP Data to Packed (or Original) Client File」は LMP 機能を利用する場合、配布されるゲームクライアントファイルにLMP情報を追加します。該当のLMP 情報は、パッキング前のオリジナルの実行ファイルにも追加できますし、パッキング後の実行ファイルにも追加できます。

注意.

そのファイルがパッキングされている場合は、“execute clientfile” を選択してください。

9. HSB File Path 入力ボックスへ AntiCpX.hsb ファイルが生成される Full Path を設定します。[...] ボタンをクリックするとファイルを生成するウィンドウが開きます。(参考：実行ファイルと同じ場所に設定すると、クライアントファイルとともに配布されることがあるとの警告メッセージが表示されます。これは正しいメッセージです。)

10. すべての設定が終わったら「Generate」ボタンをクリックします。進行状況がプログレスバーに表示され、完了したら正常に完了したとメッセージがでます。

anticpx.hsb ファイルが生成されたら anticpx.hsb ファイルをサーバーに適用した後、anticpx.hsb ファイルの生成に使うゲームクライアントファイルにもパッチを適用します。

HSB 情報ファイルの自動生成

HSBGen は HSB ファイル生成に必要な引数情報を指定することで HSB ファイルを自動的に生成します。使用方法は下記のとおりです。:

使用方法:

HSBGen.exe (1),(2),(3),(4),(5),(6),(7),(8)

- (1): HSB 情報を抽出するゲームのパス
- (2): パッキングされた実行ファイルのサポートの有無
(1 = サポート有 / 0 = サポート無)
- (3): HSB 情報を抽出するパッキングされたゲームのパス
(2)の値が 1 である場合にのみ使用)
- (4): パッキングされた実行ファイルの実行有無(1 = 実行 / 0 = 実行しない)
- (5): パッキングされた実行パラメーター ((4)の値が 1 である場合にのみ使用)
- (6): HSB ファイルパス
- (7): 保護対象ファイルのタイプ (1= EXE , 0 = DLL)
- (8): LMP 情報追加有無 (1 = 追加, 0 = 追加しない)

注意

引数情報は‘ , ’(カンマ)で区切ります。

例 1:> 以下場合の例です。:
パッキングされていないゲームの場合、
ゲームのパス: C:¥HS_SDK¥Game.exe
HSB ファイルのパス: C:¥HS_SDK¥AntiCrack¥anticpx.hsb.



図. 9-4 Command-line Type HSBGen.exe (一般ファイルの場合)

例 2:> 以下場合の例です。:

パッキングされているゲームの場合、

パッキング前のオリジナルのゲームのパス: C:\HS_SDK\Game.exe

パッキング後のゲームのパス: C:\HS_SDK\Game_packed.exe

HSB ファイルのパス: C:\HS_SDK\AntiCrack\anticpx.hsb.



図. 9-5 Command-line Type HSBGen.exe (パッキングされたファイルの場合)

例 3:> 以下場合の例です。:

パッキングされたゲームを実行する場合、

パッキング前のオリジナルのゲームのパス: C:\HS_SDK\Game.exe,

パッキング後のゲームのパス: C:\HS_SDK\Game_packed.exe

HSB ファイルのパス: C:\HS_SDK\AntiCrack\anticpx.hsb

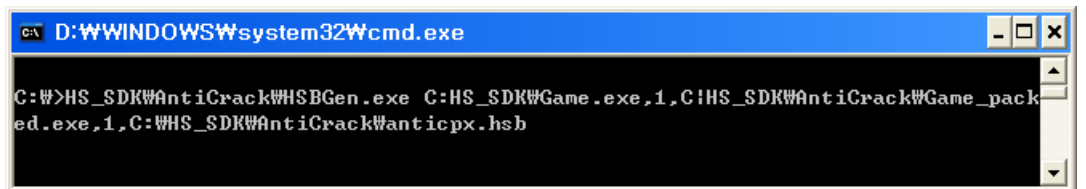


図. 9-6 Command-line Type HSBGen.exe (パッキングされたファイルを実行する場合)

例4:> 以下場合の例です。:

パッキングされたゲームを実行する時、引数が必要な場合、

パッキング前のオリジナルのゲームのパス: C:\HS_SDK\Game.exe

パッキング後のゲームのパス: C:\HS_SDK\Game_packed.exe

HSB ファイルのパス: C:\HS_SDK\AntiCrack\anticpx.hsb

ファイル実行関連引数: 210.0.0.1:7877



```
C:\WINDOWS\system32\cmd.exe
C:\W>HS_SDK\AntiCrack\HSBGen.exe C:\WHS_SDK\Game.exe,1,C:\WHS_SDK\AntiCrack\Game_packed.exe,1,210.0.0.1:7877,C:\WHS_SDK\AntiCrack\anticpx.hsb
```

図. 9-7 Command-line Type HSBGen.exe (パッキングされたファイル実行時、引数が必要な場合)

例 5:> 以下場合の例です。:

パッキングされたゲームを実行する場合、

パッキング前のオリジナルのゲームのパス: C:\WHS_SDK\Game.exe

パッキング後のゲームのパス: C:\WHS_SDK\Game_packed.exe

HSB ファイルのパス: C:\WHS_SDK\AntiCrack\anticpx.hsb

保護ファイルタイプ: exe(=1)

LMP 情報を: 追加(=1)



```
C:\WINDOWS\system32\cmd.exe
C:\W>HS_SDK\AntiCrack\HSBGen.exe C:\WHS_SDK\Game.exe,1,C:\WHS_SDK\AntiCrack\Game_packed.exe,1,210.0.0.1:7877,C:\WHS_SDK\AntiCrack\anticpx.hsb,1,1
```

例 6:> 以下場合の例です。:

パッキングされていないDLLファイルに適用する場合、


パッキング前の DLL のパス: C:\WHS_SDK\Game.dll and

HSB ファイルのパス: C:\WHS_SDK\AntiCrack\anticpx.hsb

保護ファイルタイプ: DLL(=0)

. LMP 情報: 追加(=1)

(LMP 2.0 の情報は保護対象ファイルタイプがDLLである場合、デフォルトで追加されます。)



```
C:\WINDOWS\system32\cmd.exe
C:\W>C:\WHS_SDK\AntiCrack\HSBGen.exe C:\WHS_SDK\Game.dll,0,C:\WHS_SDK\AntiCrack\anticpx.hsb,0,1
```

HSBGen.ini 説明

[VERCT]

GrantOldSession=1
MaxAllowedNumber=1

[REQRE]

InitStep1=1
InitStep2=2
InitStep3=8
InitStep4=4

[DELAY]

UseDelayedSpiking=0
MinDelayCount=1
MaxDelayCount=3

[CKMEM]

PageGroupSize=40
QueryPages=10

[FPATH]

ClientFileName=D:¥ ¥game_ori.exe
UsePackedClientFile=1
PackedClientFileName=D:¥ ¥game_packed.exe
GetInfoFromRunningEXE=1
Parameters=-d
AppendLMPInfoToClientFile=1
HsbFileName=D:¥ ¥anticpx.hsb
UseEXE=0

[CKHSB]

UseHSB=1

1. VERCT

GrantOldSession

上記の値は1または0を使用します。‘1’の場合現在のhsbファイルにあうクライアントバージョン以外にも、以前のhsbファイルに対するクライアントバージョンを許可します。

MaxAllowedNumber

GrantOldSession が‘1’の場合だけ有効です。1以上の値を与える必要があります。‘1’の場合には現在適用されたhsbに対するクライアントのみ許可されます。‘5’の場合には以前に適用したhsbファイルに対するクライアントバージョンを4つまで許可します。

2. REQRE

サーバー連動をする際のスキャン機能は、GUIDスキャン、クライアントファイルスキャン、HackShieldモジュールスキャン、メモリスキャン、HackShield動作状態スキャンの5つがあります。

GUID スキャン:	1
クライアントファイル スキャン:	2
メモリ スキャン:	4
HackShield エンジンスキャン:	8
HackShield 動作状態スキャン:	16

※’ HackShield 動作状態スキャン’ 機能は初期設定からすべてのステップ(InitStep1, InitStep2, InitStep3, InitStep4) に対して適用されます

各スキャンオプションは、1, 2, 4, 8, 16 bitが存在し重複使用可能です。

InitStep 1

GUID scan(1) のみをスキャン、またはGUID scan(1)と HackShield 動作状態スキャン(16) の同時スキャン(17)のみ可能です。

InitStep 2

GUID scan(1) およびクライアントファイルスキャン(2)を使用可能です。

InitStep 3

すべてのスキャンを使用可能です。

‘31’の値を与えた場合、すべてのスキャンを使用します。‘5’を与えた場合、メモリスキャンとHackShieldモジュールスキャンを使用します。

InitStep 4

上記3つのステップが完了し、周期的に反復するスキャンを指定します。
通常メモリスキャンおよびHackShield動作スキャンオプションであり‘20’を使用します。

3. DELAY

UseDelayedSpiking

‘1’を使用して機能を使用する場合、_AhnHS_VerifyResponse 関数からエラーをリターンした時にすぐエラーを発生させず一定のデレイを発生させ、ハッカーに混乱を与えます。
‘0’の場合、上記機能を使用しません。

MinDelayCount, MaxDelayCount

ディレイスパイキング機能が適用された場合、ディレイはMinDelayCountとMaxDelayCountの間のランダムな値が与えられます。1は関数が呼び出される一つの周期です。

4. CKMEM

PageGroupSize, QueryPages

メモリスキャン時に、一度にスキャン可能なメモリページ数を指定します。初期値で使用することをお勧めします。

5. FPATH

HSBGen.exeツールを使用してユーザーが入力した情報で、ゲームクライアントのメモリCRC情報ファイルである.hsbファイル生成のためのゲームクライアント設定情報です。

ClientFileName

パッキングされていないゲームクライアントのパスです

UsePackedClientFile

パッキングしたゲームクライアントの使用有無に対する設定です。

パッキングしたゲームクライアントを使用する場合、この値は1であり、パッキングしたゲームクライアントを使用しない場合、値は0となります。

PackedClientFileName

パッキングされたゲームクライアントのパスです。

UsePackedClientFile の値が1の場合には必要です。

GetInfoFromRunningEXE

パッキングしたゲームクライアントを使用する場合、ゲームクライアントを実行させ、メモリCRC情報を抽出するための設定値です。

この値が1の場合、PackedClientFileName.に設定されたパスのファイルを実行させメモリCRC情報を抽出します。

Parameters

ゲームクライアント実行時に必要なパラメータ情報です。PackedClientFileNameにパスが設定され、GetInfoFromRunningEXEの値が1の場合に有効です。

AppendLMPInfoToClientFile

LMP機能を使用するために必要な関連情報をゲームクライアントファイルに追加するための設定値です。

値が1の場合LMP機能を使用するために必要な情報をゲームクライアントファイルに追加します。値が0の場合、作業を実行しません。

HsbFileName

ゲームクライアントのメモリCRC情報ファイル(.hsbファイル)が生成されるパスです。この値を設定しないと、.hsbファイルを生成できません。

(必ず拡張子は.hsbで保存してください。)

UseEXE

対象ファイルがEXEの場合、1、DLLの場合0を設定します。

注意.

SDKにある基本hsbgen.iniファイルをopenした場合、FPATHパスが存在しません。hsbファイルを生成した時にはじめてFPATH情報が保存されます。

6. CKHSB

UseHSB

新規ゲームクライアントファイルを配布した時、ゲームクライアントファイルを対象にしてHSSBGen.exeツールを実行して生成したHSBファイルをゲームサーバーにアップロードするかを設定します。

値が0(UseHSB=0)の場合、'新規ゲームクライアントファイルを対象に生成されたHSBファイルをゲームサーバーへアップロードせず拡張サーバーと連動'します。

値が1(UseHSB=1)の場合、'新規ゲームクライアントファイルを対象に生成されたHSBファイルをゲームサーバーへアップロードし拡張サーバーと連動'します

セクション内の UseHSB=0 の場合、機能使用時の注意事項

上記セクションの内容が適用されたHSBファイル(最初一回)を必ずサーバーにアップロードしてください。

新規ゲームクライアントファイル配布前にHSBGenツールを使用して正常にHSBファイルが生成されたか確認する必要があります。(ただし、HSBファイルをゲームサーバーにアップロードする必要はありません)

HSBファイル生成時にHSBGen.iniファイル内の内容が、ゲームクライアントに反映されるためHSBGen.iniファイルの内容の確認を行ってください

'クライアントファイルスキャン'、'メモリスキャン' はサポートしません

[REQRE] セクション内の'InitStep2'、'InitStep3'、'InitStep4' 内に'クライアントファイルスキャン' 値および'メモリスキャン' の値を意味する数値は'2'、'4'の値ははずして適用する必要があります

例> InitStep1=1
InitStep2=1
InitStep3=8
InitStep4=1

注意

HSBGen.exe ツール使用して新たに生成したHSBファイルの内容が適用されるのはHSBファイルの生成時に入力されたゲームクライアントファイルがサーバーに接続した後です。

つまり、新たに生成されたHSBファイルをゲームサーバーにアップロードしたとしても反映されることはありません。HSBファイルの生成時に入力した該当のゲームクライアントで最初一回はゲームサーバーに接続する必要があります。

9.5. HSUpSetEnv ツール(HackShield専用, 5.1以後バージョン)

機能

アップデート環境設定ファイル(HSUpdate.env)を作成・更新する。
FTP、HTTP を選択可能、また複数の URL を設定可能。

9.6. HSUpSetEnvツールの使い方

HSUpdate.envファイル作成

[HackShield SDK]¥Bin¥Win¥x86¥Util¥HSUpSetEnv.exe を利用して HSUpdate.env ファイルを作成します。

[General information] と [Extended information] の2つのタブから構成されます。:

使用方法:

[HackShield SDK]¥Bin¥Win¥x86¥Util¥HSUpSetEnv.exe ファイルを実行します。

注意

HSUpSetEnv.exe を配布することはライセンス上できません。

HackShield Update Configuration

Basic Information Extended Information

Server Connect configuration

Number of Maximum Retries : [1] (6)

Update Server Information Configuration

☐ HTTP ☒ FTP (1)

Server Name : [] (2)

Server Address : [] (3) [] (4)

User ID : []

Password : [] (5)

Name	Protocol	IP/URL	Port	ID	PWD
------	----------	--------	------	----	-----

Add (7) Delete (8)

Save (14) Exit

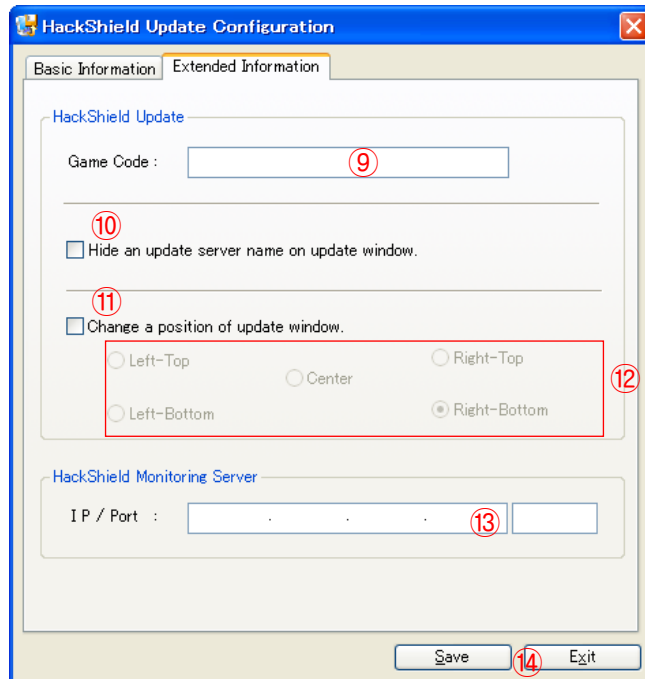
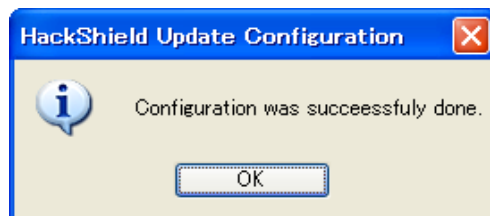


図. 9-8 HSUpSetEnv.exe

- ① HTTP または、FTP を選択します。
- ② アップデートサーバー名を入力します。
- ③ URLを入力します。(パッチセットがあるフォルダのパスまで)
- ④ ポート番号を入力します。
- ⑤ FTPの場合、アカウントを入力します。(入力しない場合、anonymous接続になる)
- ⑥ リトライ回数を指定します。
- ⑦ サーバーを追加することができます。
- ⑧ サーバーを削除します。
- ⑨ 有効なゲームコードを入力します。
- ⑩ チェックすると、アップデートイメージにサーバーアドレスを表示しません。
- ⑪ アップデートイメージの表示位置を変更できます。
- ⑫ アップデートイメージの表示位置を指定します。
- ⑬ モニタリングサーバーのIP、ポートを指定します。
- ⑭ 保存、または 終了します。



設定が終了したら、ツールを終了します。

9.7. CSInspectorツール

機能

特定のEXEファイル、DLL ファイルを保護対象に設定します。このツールはパッキングする前に実行する必要があります。サーバー連動機能と共に使う場合は、AntiCpSvrTool.exe または HSBGen.exe よりをより先に実行する必要があります。現在は、コマンドライン形式のみの提供です。

CSInspector は、クライアントでAHNHS_CHKOPT_LOCAL_MEMORY_PROTECTIONと共に利用する必要があります。

対象のファイルを保護対象に設定

¥Bin¥Win¥x86¥Util¥CSInspector.exeを使って特定のEXEファイル、DLL ファイルを保護対象に設定するためには、以下を実行します。:

使用方法:

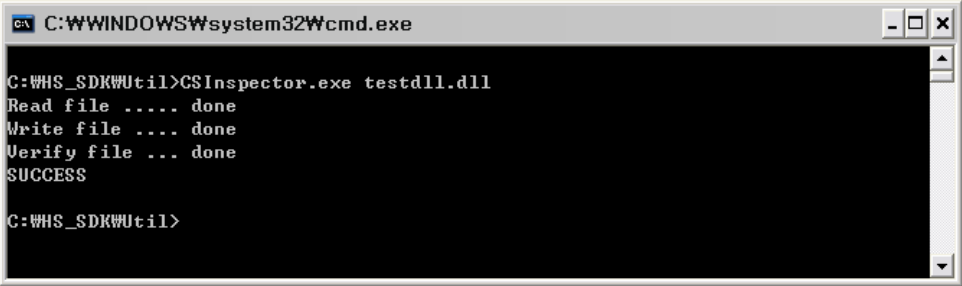
CSInspector.exe [対象のファイル]

注意

パッキングまたは、CRC生成を行う前に実行してください。

例:

対象ファイルが testdll.dll で、CSInspector.exe が C:¥HS_SDK¥Util folder にある場合の例です。:



```
C:\WINDOWS\system32\cmd.exe

C:\HS_SDK\Util>CSInspector.exe testdll.dll
Read file ..... done
Write file .... done
Verify file ... done
SUCCESS

C:\HS_SDK\Util>
```

図. 9-9 CSInspector.exe

9.8. SetServerListツール(HackShield専用, 5.1以後バージョン)

機能

ANTIFREESERVER 機能を利用するために、SetServerList を利用して有効な IP アドレスを設定する必要があります。ANTIFREESERVERオプションを使った状態で SetServerListでIPアドレスを設定した場合、ゲームのプロセスで指定されていない IP アドレスへのアクセスやローカルホスト(127.0.0.1) へアクセスを試みると、コールバックが発生します。

この機能を利用する場合、クライアントで AHNHS_CHKOPT_ANTIFREESERVER オプションを指定する必要があります。

9.9. SetServerList ツールの使い方

afs.datファイル作成

[HackShield SDK]¥Bin¥Win¥x86¥Util¥SetServerList.exe を利用して afs.dat を作成します。以下の手順で利用します。:

使用方法:

[HackShield SDK]¥Bin¥Win¥x86¥Util¥SetServerList.exe ファイルを実行します。

注意

SetServerList.exe はライセンス上配布することができません。.

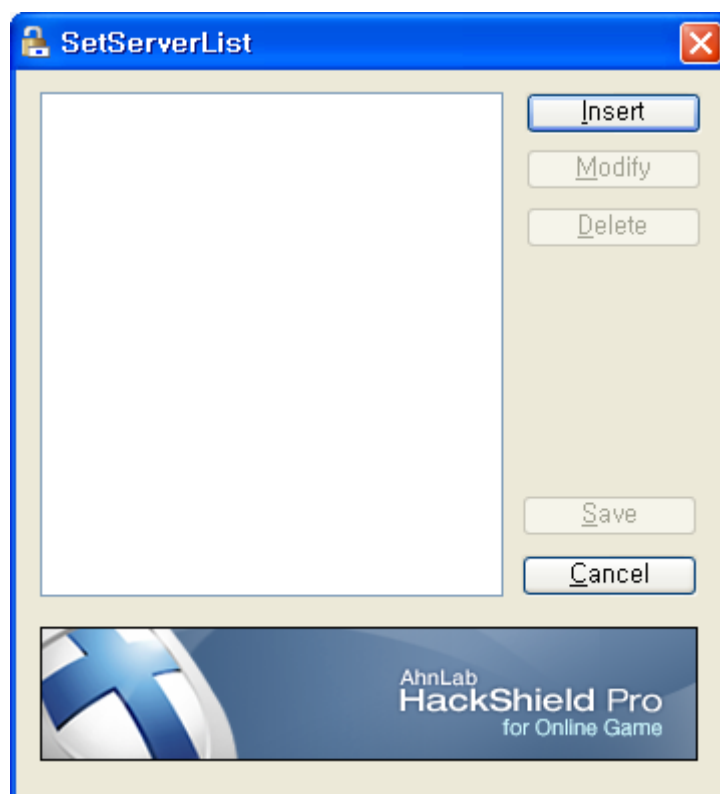


図. 9-10 SetServerList.exe

- ① [Insert]ボタンを押して IP アドレスを入力します。

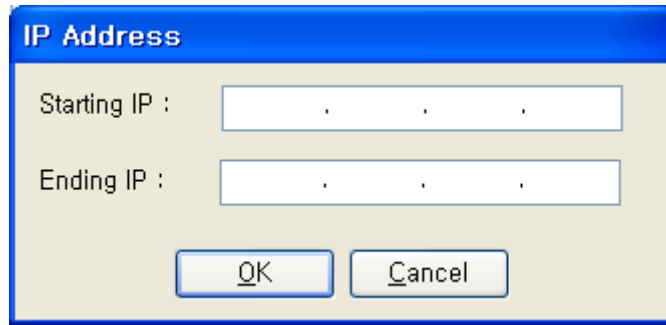


図. 9-11 SetServerList ツールのIPアドレス入力ウィンドウ

- ② 有効な IP アドレスの範囲を指定します。1つのみ入力したい場合は、Starting IP と Ending IP に同じ値を入力します。
- ③ OK をクリックして 2 で入力したアドレスを追加します。
- ④ IP アドレスをダブルクリックするか、Modify をクリックして変更することができます。
- ⑤ Delete をクリックすると IP アドレス を削除することができます。
- ⑥ Save をクリックすると設定した内容で afs.dat ファイルを作成します。
他の方法で終了すると、編集内容が失われます。
- ⑦ afs.dat ファイルを配布するときは、HackShield モジュールと同じ場所に
設置してください。(例. [Game Directory]/hshield/afs.dat)
- ⑧ SerServerList.exe.を配布することはできません。

afs.dat ファイル配布

ゲーム会社は、afs.dat を管理して配布する必要があります。

ゲーム会社が afs.dat ファイルを配布できない場合、HackShield アップデートで配布することが可能です。

(HackShield アップデートで配布する場合にも、ゲーム会社はファイルを厳重に管理する必要があります。)

注意

HackShield アップデートについては、マニュアルを参照してください。

[HackShield アップデート機能] - [\[システムアーキテクチャ\]](#)

9.10.HSBHelperツールの使い方

機能

¥Bin¥Win¥x86¥AntiCrack¥HSBHelper を利用してHSB ファイルとそれに対応するゲームクライアントファイルが正しいものか確認します。以下の手順で利用します。:

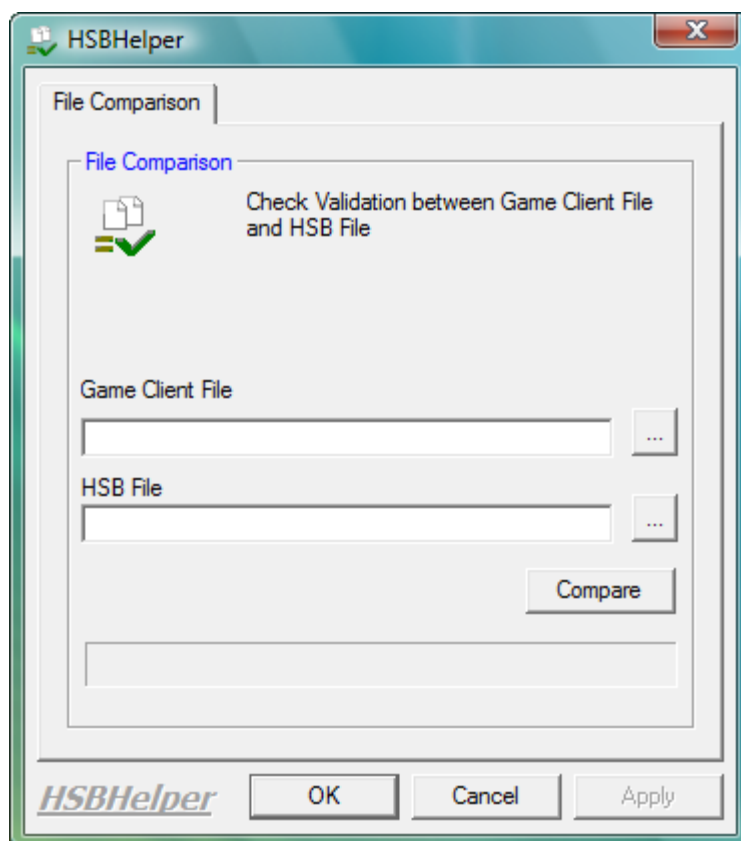
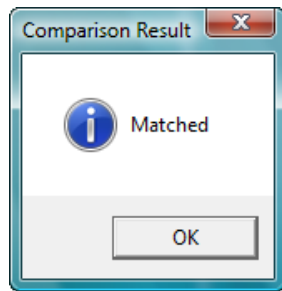


図. 9-12 HSBHelper Tool

- ① ゲームクライアントのフルパスを入力します。
(例. C:¥ R.5.1.41.1(build 671)¥Bin¥Win¥x86¥Amazon.exe)
- ② HSB ファイルのフルパスを入力します。
(例. C:¥ R.5.1.41.1(build 671)¥Bin¥Win¥x86¥AntiCrack¥AntiCpX.hsb)
- ③ [Compare] をクリックします。
- ④ ファイルがマッチすると以下の画面が表示されます。.



そうでない場合以下の画面が表示されます。



注意

HSBHelper を実行すると、ログファイル HSBHelper.log が常に作成されます。

10. 付録

10.1.FAQ

SoftICE をインストールしているシステムで HackShield の初期化が正常に動作しません。どのような設定をしたら、デバッグ可能になりますか？

開発・テストの工程では、HShield.lib, EhSvc.dll, hshield.dat の開発バージョンを利用する必要があります。

(サーバーとの連携で問題が発生する場合は、EhSvc.dll と hshield.dat のバージョンをクライアント、サーバー間で同一か確認してください。)

HackShield の開発版は以下の場所にあります。

EhSvc.dll - ¥SDK¥Korean(kr)-SDK¥Developer¥Bin¥Win¥x86¥HShield

hshield.dat - ¥SDK¥Korean(kr)-SDK¥Developer¥Bin¥Win¥x86¥HShield

(サーバー連動の適用時(後)デバッグングするためにサーバーにコピーしてください。)

HShield.lib - ¥SDK¥Korean(kr)-SDK¥Developer¥Lib¥Win¥x86¥

(ゲームクライアントプロジェクトのコンパイル項目に合わせて適切なライブラリファイルを利用してください。)

注意

VC++ コンパイラの Microsoft C++ 例外で常に停止するように設定している場合、以下のようなエラーが発生する場合があります。

“First chance exception in Game.exe(KERNEL32.dll) 0xE0607063 Microsoft C++ Corporation”

この場合、例外設定を確認して、例外をハンドルしないように設定を変更してください。

Windows Vista の マニフェスト機能を使うにはどうしたいですか？

参考 (Manifest)

Vista で動作するアプリケーションを管理者権限で動作するように作成するためには、実行ファイルのプログラム情報に権限を追加する必要があります。もともと、ユーザーは右クリックを実行すると“管理者として実行”をすることができますし、プログラムのプ

ロパティで管理者と実行を選択する方法もあります。がこれらの方法は、面倒な方法です。マニフェストを利用すると、アプリケーションに「管理者権限を必要としている」という情報を追加することで上記と同じことを、常に自動的に行うことが可能です。

ゲームのリソースにマニフェスト (xml) を適用します。

- ① Visual Studio6.0. を実行します。
- ② ゲームリソースフォルダに リソースを追加します。
- ③ 新規でカスタムリソース、リソースタイプ “24” を追加します。

参考

マニフェストリソースの “24” はマイクロソフトにより定義されています。

- ④ IDR_DEFAULT1 以下の xml を追加します。

```
<?xml version="1.0" encoding="utf-8" ?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity version="1.0.0.0"
    processorArchitecture="X86"
    name="Game"
    type="win32" />
  <description>HspL</description>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level="requireAdministrator" />
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

- ⑤ 追加すると、以下のバイナリとアスキーコードを xml ドキュメントビューで確認することができます。

000000	3C 3F 78 6D 6C 20 76 65	72 73 69 6F 6E 3D 22 31	<?xml version="1
000010	2E 30 22 20 65 6E 63 6F	64 69 6E 67 3D 22 75 74	.0" encoding="ut
000020	66 2D 38 22 20 3F 3E 0D	0A 3C 61 73 73 65 6D 62	f-8" ?>..<assemb
000030	6C 79 20 78 6D 6C 6E 73	3D 22 75 72 6E 3A 73 63	ly xmlns="urn:sc
000040	68 65 6D 61 73 2D 6D 69	63 72 6F 73 6F 66 74 2D	hemas-microsoft-
000050	63 6F 6D 3A 61 73 6D 2E	76 31 22 20 6D 61 6E 69	com:asm.v1" mani
000060	66 65 73 74 56 65 72 73	69 6F 6E 3D 22 31 2E 30	festVersion="1.0
000070	22 3E 0D 0A 3C 61 73 73	65 6D 62 6C 79 49 64 65	">..<assemblyIde
000080	6E 74 69 74 79 20 76 65	72 73 69 6F 6E 3D 22 31	ntity version="1
000090	2E 30 2E 30 2E 30 22 20	0D 0A 20 20 20 20 70 72	.0.0.0" .. pr
0000a0	6F 63 65 73 73 6F 72 41	72 63 68 69 74 65 63 74	rocessorArchitect
0000b0	75 72 65 3D 22 58 38 36	22 0D 0A 20 20 20 20 6E	ure="X86".. n
0000c0	61 6D 65 3D 22 4D 69 6E	69 41 22 0D 0A 20 20 20	ame="MiniA"..
0000d0	20 74 79 70 65 3D 22 77	69 6E 33 32 22 20 2F 3E	type="win32" />
0000e0	20 0D 0A 20 20 3C 64 65	73 63 72 69 70 74 69 6F	.. <descriptio
0000f0	6E 3E 4D 69 6E 69 41 3C	2F 64 65 73 63 72 69 70	n>MiniA</descrip
000100	74 69 6F 6E 3E 0D 0A 20	20 3C 74 72 75 73 74 49	tion>.. <trustI
000110	6E 66 6F 20 78 6D 6C 6E	73 3D 22 75 72 6E 3A 73	nfo xmlns="urn:s
000120	63 68 65 6D 61 73 2D 6D	69 63 72 6F 73 6F 66 74	chemas-microsoft
000130	2D 63 6F 6D 3A 61 73 6D	2E 76 33 22 3E 0D 0A 20	-com:asm.v3">..
000140	20 20 20 3C 73 65 63 75	72 69 74 79 3E 0D 0A 20	<security>..
000150	20 20 20 20 20 3C 72 65	71 75 65 73 74 65 64 50	<requestedP
000160	72 69 76 69 6C 65 67 65	73 3E 0D 0A 20 20 20 20	rivileges>..
000170	20 20 20 20 20 20 3C 72	65 71 75 65 73 74 65 64	<requested
000180	45 78 65 63 75 74 69 6F	6E 4C 65 76 65 6C 20 6C	ExecutionLevel l
000190	65 76 65 6C 3D 22 72 65	71 75 69 72 65 41 64 6D	evel="requireAdm
0001a0	69 6E 69 73 74 72 61 74	6F 72 22 20 20 2F 3E 0D	inistrator" />.
0001b0	0A 20 20 20 20 20 20 3C	2F 72 65 71 75 65 73 74	. </request
0001c0	65 64 50 72 69 76 69 6C	65 67 65 73 3E 0D 0A 20	edPrivileges>..
0001d0	20 20 20 3C 2F 73 65 63	75 72 69 74 79 3E 0D 0A	</security>..
0001e0	20 20 3C 2F 74 72 75 73	74 49 6E 66 6F 3E 0D 0A	</trustInfo>..
0001f0	3C 2F 61 73 73 65 6D 62	6C 79 3E	</assembly>

⑥ IDR_DEFAULT1 を右クリックして、プロパティをクリックします。

⑦ IDR_DEFAULT1 を 1 に変更します。

参考

24 マニフェストを利用するには、ID は 1 でないといけません。

⑧ 保存し、ゲームのリソースをリビルドします。

参考（なぜマニフェストを利用する必要があるのか？）

ハッカーまたは、悪意のあるユーザーが管理者権限でハッキングツールを実行した場合、ゲームが相対的に弱い権限で動作することになるのでハッキング攻撃に対し無防備になります。また、Vista は実行中のプロセスが権限を昇格することができません。

管理者権限を持つハッキングツールから保護するために、HackShield は管理者権限で動作しなければなりません。Vista ではシャドウアカウントは必須ではありませんが、マニフェストは必須です。

Windows 2000 や Windows XP で管理者権限のないユーザーでログインし、HackShield を動作させることはできますか？

HackShield のハッキング遮断ドライバはカーネルレベルで動作します。だから、基本的な設定では HackShield を動作させるためには、管理者権限が必要です。通常のアカ

ウントで HackShield を動作させるには、シャドウアカウントを作成する必要があります。

ゲーム サービス 運用中に、HackShield 終了関数（AhnHS_StopService、_AhnHS_Uninitialize）が呼ばれることなく、ゲームが異常終了してしまいました。ゲームを再起動することで HackShield を再度動作させることはできますか？

ゲームが HackShield 終了関数を呼び出すことなく終了してしまった場合、HackShield ハッキング遮断ドライバがアンロードされません。ゲームを再度起動してHackShield初期化かんすうがを呼び出した場合は、ロード済みのドライバは、強制的にアンロードされ、新規にロードしなおします。その後、通常通りの初期化プロセスを実行します。

新たなハッキングツールが見つかった場合、どのように防御できますか？

新たなハッキングツールが見つかった場合には、以下の情報をアンラボに送信してください。

アンラボは、ハッキングツールの情報を解析して、その結果エンジンアップデートに反映します。(定期アップデート、および緊急アップデート)

ハッキングツールの対象ゲームタイトル

OS version

ハッキングツールにかんする簡単な内容

ハッキングツール本体