

# AhnLab

## HackShield for Online Game 2.0



프로그래밍 가이드

# 일러두기

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

이 프로그래밍 가이드의 내용과 프로그램은 저작권법과 컴퓨터프로그램보호법에 의해서 보호받고 있습니다.

## 면책 조항

제조자, 수입자, 대리점은 상해를 포함하는 우발적인 손상 또는 본 제품의 부적절한 사용과 조작으로 인한 기타 손상에 대해 책임을 지지 않습니다.

이 프로그래밍 가이드의 내용은 현재 제품을 기준으로 작성되었습니다. (주)안랩은 지금도 새로운 기능을 추가 보완하고 있으며 향후에도 지속적으로 새로운 기술을 적용할 것입니다. 제품의 모든 기능은 제품 구입자 또는 제품 구입 기업에게 사전 통보 없이 변경될 수 있으며 이 프로그래밍 가이드의 내용과 차이가 날 수 있습니다.

## 표기 규칙

표기 규칙은 다음과 같습니다.

표기 규칙	내용
abcd, 가나다라	프로그래밍에 사용된 코드 내용
 참고	프로그램을 사용할 때 참고할 사항
 주의	프로그램을 사용할 때 주의해야 할 사항
HackShield	AhnLab HackShield for Online Game 2.0의 줄임말
고객	고객(게임 퍼블리셔), 게임 개발사 등 HackShield 사용 대상의 통칭
게임 유저	HackShield의 고객이 제공하는 게임 서비스를 이용하는 일반 사용자

## 가이드 구성

HackShield 가이드는 다음과 같이 4개로 나누어져 있습니다. 이 문서는 SDK 형태로 제공되는 HackShield의 전체 구조와 기능 및 API 함수 사용 방법을 설명한 프로그래밍 가이드입니다.

- AhnLab HackShield for Online Game 2.0 서버 퀵 가이드
- AhnLab HackShield for Online Game 2.0 클라이언트 퀵 가이드
- AhnLab HackShield for Online Game 2.0 업데이트 퀵 가이드
- AhnLab HackShield for Online Game 2.0 프로그래밍 가이드

## 고객 지원

---

고객 지원 연락처는 다음과 같습니다.

- 주소: 463-400 경기도 성남시 분당구 삼평동 673 번지 (주)안랩
- 홈페이지: <http://www.ahnlab.com>
- 메일 주소: [hs\\_support@ahnlab.com](mailto:hs_support@ahnlab.com)
- 전화: 1577-9431
- 팩스: 031-722-8901

# 목차

---

<b>1장 HackShield 소개</b>	<b>11</b>
1.1    HackShield 소개	12
1.2    시스템 요구 사항	12
<b>2장 기본 기능</b>	<b>14</b>
2.1    기본 기능	15
2.1.1    기능 소개	15
2.1.2    기능 특징	16
2.1.3    시스템 구조	17
2.2    애플리케이션 프로그래밍	18
2.2.1    프로그래밍 순서	18
2.2.2    프로그래밍 준비	20
2.2.3    프로그래밍 적용	22
2.3    애플리케이션 프로그래밍 인터페이스	31
2.3.1    _AhnHS_Initialize()	31
2.3.2    _AhnHS_Callback()	43
2.3.3    _AhnHS_StartService()	49
2.3.4    _AhnHS_StopService()	51
2.3.5    _AhnHS_Uninitialize()	52
2.3.6    _AhnHS_PauseService()	52
2.3.7    _AhnHS_ResumeService()	53
2.3.8    _AhnHS_CheckHackShieldRunningStatus()	55
2.3.9    _AhnHS_SendHsLog()	56
2.3.10    _AhnHS_VerifyProtectedFunction()	56
2.3.11    _AhnHS_ThreadStart	58
2.3.12    _AhnHS_ThreadStop	60
2.3.13    _AhnHS_ThreadStartEx()	61
2.3.14    _AhnHS_ThreadStopEx	64
2.3.15    _AhnHS_SetProtectedFunction	65
2.3.16    _AhnHS_CheckProtectedStatus()	67
2.3.17    _AhnHS_QueryPerformanceCounter()	70
2.3.18    _AhnHS_QueryPerformanceFrequency()	72
2.3.19    _AhnHS_GetTickCount()	73
<b>3장 HackShield 업데이트 기능</b>	<b>76</b>
3.1    HackShield 업데이트 기능	77
3.1.1    기능 소개	77
3.1.2    기능 특징	77

3.1.3	시스템 구조 .....	78
<b>3.2</b>	<b>애플리케이션 프로그래밍 .....</b>	<b>79</b>
3.2.1	프로그래밍 준비 .....	79
3.2.2	프로그래밍 적용 .....	83
<b>3.3</b>	<b>애플리케이션 프로그래밍 인터페이스 .....</b>	<b>88</b>
3.3.1	_AhnHS_HSUpdateEx() .....	88
3.3.2	_AhnHS_HSUpdate() .....	93
<b>4장 서버 연동 크랙 방지 기능 .....</b>		<b>97</b>
<b>4.1</b>	<b>서버 연동 크랙 방지 기능 .....</b>	<b>98</b>
4.1.1	기능 소개 .....	98
4.1.2	기능 특징 .....	99
4.1.3	시스템 구조 .....	99
<b>4.2</b>	<b>애플리케이션 프로그래밍 .....</b>	<b>103</b>
4.2.1	프로그래밍 준비 .....	103
4.2.2	프로그래밍 적용 .....	104
<b>4.3</b>	<b>애플리케이션 프로그래밍 인터페이스 .....</b>	<b>106</b>
4.3.1	_AhnHS_CreateServerObject() .....	107
4.3.2	_AhnHS_CloseServerHandle() .....	108
4.3.3	_AhnHS_CreateClientObject() .....	108
4.3.4	_AhnHS_CloseClientHandle() .....	109
4.3.5	_AhnHS_MakeRequest() .....	110
4.3.6	_AhnHS_VerifyResponseEx() .....	112
4.3.7	_AhnHS_VerifyResponseEx_WithInfo() .....	114
4.3.8	_AhnHS_VerifyResponse() .....	117
4.3.9	_AhnHS_MakeResponse() .....	122
<b>5장 모니터링 서비스 기능 .....</b>		<b>126</b>
<b>5.1</b>	<b>모니터링 서비스 기능 .....</b>	<b>127</b>
5.1.1	기능 소개 .....	127
5.1.2	기능 특징 .....	127
5.1.3	시스템 구조 .....	128
<b>5.2</b>	<b>애플리케이션 프로그래밍 .....</b>	<b>129</b>
5.2.1	프로그래밍 준비 .....	129
5.2.2	프로그래밍 적용 .....	129
<b>5.3</b>	<b>애플리케이션 프로그래밍 인터페이스 .....</b>	<b>131</b>
5.3.1	_AhnHS_StartMonitor() .....	131
5.3.2	_AhnHS_SetUserId() .....	132
5.3.3	_AhnHS_SetUserCustomInfo() .....	133
5.3.4	_AhnHS_SendUserCustomInfo() .....	134
5.3.5	_AhnHS_UpdateMonitorInfo() .....	136

<b>6장 LMP 기능 .....</b>	<b>138</b>
<b>6.1 Local Memory Protection(LMP) 기능 .....</b>	<b>139</b>
6.1.1     기능 소개 .....	139
6.1.2     기능 특징 .....	140
6.1.3     시스템 구조 .....	140
<b>6.2 애플리케이션 프로그래밍 .....</b>	<b>141</b>
6.2.1     프로그래밍 준비 .....	141
6.2.2     프로그래밍 적용 .....	142
<b>6.3 애플리케이션 프로그래밍 인터페이스 .....</b>	<b>144</b>
6.3.1     _AhnHS_IsModuleSecure()	144
<b>7장 부가 기능 (데이터 파일/메시지 암호화) .....</b>	<b>146</b>
<b>7.1 데이터 파일/메시지 암호화 기능 .....</b>	<b>147</b>
7.1.1     기능 소개 .....	147
7.1.2     기능 특징 .....	147
7.1.3     시스템 구조 .....	147
<b>7.2 애플리케이션 프로그래밍 .....</b>	<b>148</b>
7.2.1     프로그래밍 순서 .....	148
7.2.2     프로그래밍 준비 .....	149
<b>7.3 애플리케이션 프로그래밍 인터페이스 .....</b>	<b>150</b>
7.3.1     _HsCrypt_InitCrypt()	150
7.3.2     _HsCrypt_GetEncMsg()	151
7.3.3     _HsCrypt_GetDecMsg()	153
7.3.4     _HsCrypt_FRead()	154
<b>8장 부가 기능 (User 권한 실행 지원) .....</b>	<b>157</b>
<b>8.1 User 권한 실행 지원 기능 .....</b>	<b>158</b>
8.1.1     기능 소개 .....	158
8.1.2     기능 특징 .....	158
8.1.3     시스템 구조 .....	158
<b>8.2 애플리케이션 프로그래밍 .....</b>	<b>159</b>
8.2.1     프로그래밍 순서 .....	160
8.2.2     프로그래밍 준비 .....	161
<b>8.3 애플리케이션 프로그래밍 인터페이스 .....</b>	<b>162</b>
8.3.1     _AhnHSUserUtil_CreateUser()	162
8.3.2     _AhnHSUserUtil_SetFolderPermission()	164
8.3.3     _AhnHSUserUtil_DeleteUser()	166
8.3.4     _AhnHSUserUtil_IsEnableHSAdminRights()	166
8.3.5     _AhnHSUserUtil_CheckHSShadowAccount()	167
8.3.6     _AhnHSUserUtil_IsAdmin()	168
<b>9장 툴 사용 방법 .....</b>	<b>170</b>

<b>9.1</b>	<b>HSBGen 툴 .....</b>	<b>171</b>
9.1.1	시스템 요구 사항.....	171
9.1.2	툴 사용 방법.....	171
<b>9.2</b>	<b>HSUpSetEnv 툴 .....</b>	<b>182</b>
9.2.1	툴 사용 방법.....	182
<b>9.3</b>	<b>SetServerList 툴 .....</b>	<b>185</b>
9.3.1	툴 사용 방법.....	185
<b>9.4</b>	<b>HSBHelper 툴 .....</b>	<b>187</b>
9.4.1	툴 사용 방법.....	187
<b>10장</b>	<b>자주 하는 질문.....</b>	<b>190</b>

## 그림 목차

---

그림 1 HackShield 전체 구조 .....	17
그림 2 애플리케이션 프로그래밍 순서 .....	20
그림 3 _AhnHS_Initialize() 호출 예제.....	25
그림 4 _AhnHS_StartService() 호출 예제 .....	26
그림 5 콜백 함수 호출 예제.....	28
그림 6 _AhnHS_StopService() 호출 예제.....	29
그림 7 _AhnHS_Uninitialize() 호출 예제 .....	29
그림 8 HackShield 로그 전송 .....	30
그림 9 AHNHS_CHKOPT_SELF_DESTRUCTION 옵션 예제 .....	36
그림 10 AHNHS_SPEEDHACK_EXOPT_ALLOW_SLOW 사용 예제 .....	39
그림 11 SELF-DESTRUCTION TIME RATIO 사용 예제 .....	40
그림 12 HackShield 로고 화면 .....	43
그림 13 _AhnHS_CheckProtectedStatus()를 통해 요청 가능한 타입 .....	67
그림 14 _AhnHS_CheckProtectedStatus()의 정보 전달 구조체 .....	68
그림 15 HackShield 업데이트 동작 원리.....	78
그림 16 AntiCpX의 동작 원리 .....	100
그림 17 서버 연동 주기 .....	106
그림 18 _AhnHS_CreateServerObject() 호출 예제.....	108
그림 19 _AhnHS_CloseServerHandle() 호출 예제.....	108
그림 20 _AhnHS_CreateClientObject() 호출 예제 .....	109
그림 21 _AhnHS_CloseClientHandle() .....	110
그림 22 _AhnHS_MakeRequest() 호출 예제 .....	112
그림 23 _AhnHS_VerifyResponseEx() 호출 예제 .....	114
그림 24 _AhnHS_VerifyResponseEx_WithInfo() 호출 예제.....	117
그림 25 _AhnHS_VerifyResponse() 호출 예제 .....	122
그림 26 _AhnHS_MakeResponse() 사용 예제.....	125
그림 27 모니터링 서비스의 동작 원리.....	128
그림 28 _AhnHS_StartMonitor() 호출 예제 .....	132
그림 29 _AhnHS_SetUserId() 호출 예제 .....	133
그림 30 _AhnHS_SetUserCustomInfo() 호출 예제.....	134
그림 31 _AhnHS_SendUserCustomInfo() 호출 예제.....	136
그림 32 _AhnHS_UpdateMonitorInfo() 호출 예제.....	137
그림 33 Local Memory Protection 동작 구조 .....	140
그림 34 _AhnHS_IsModuleSecure() 호출 예제 .....	144
그림 35 _AhnHS_IsModuleSecure() 호출 예제 .....	145
그림 36 HsCryptLib의 전체 구조 및 동작 원리 .....	148

그림 37	_HsCrypt_InitCrypt() 호출 예제 .....	151
그림 38	_HsCrypt_GetEncMsg() 호출 예제.....	153
그림 39	_HsCrypt_GetDecMsg() 호출 예제 .....	154
그림 40	_HsCrypt_FRead() 호출 예제 .....	156
그림 41	HsUserUtil의 전체 구조 및 동작 원리 .....	159
그림 42	HsUserUtil 프로그래밍 순서 .....	161
그림 43	_AhnHsUserUtil_CreateUser() 호출 예제.....	163
그림 44	_AhnHsUserUtil_SetFolderPermission() 호출 예제 .....	165
그림 45	_AhnHsUserUtil_DeleteUser() 호출 예제.....	166
그림 46	_AhnHsUserUtil_IsEnableHSAdminRights() 호출 예제 .....	167
그림 47	_AhnHsUserUtil_CheckHSShadowAccount() 호출 예제 .....	168
그림 48	_AhnHSUserUtil_IsAdmin() 호출 예제 .....	169
그림 49	Command-line 방식의 HSBGen.exe ([패킹:1],[실행:1], [EXE:1], [LMP:1]) .....	174
그림 50	Command-line 방식의 HSBGen.exe ([패킹:1],[실행:1], [EXE:1], [LMP:0]) .....	175
그림 51	Command-line 방식의 HSBGen.exe ([패킹:1],[실행:0], [EXE:1], [LMP:1]) .....	175
그림 52	Command-line 방식의 HSBGen.exe ([패킹:1],[실행:0], [EXE:1], [LMP:0]) .....	175
그림 53	Command-line 방식의 HSBGen.exe ([패킹:0],[실행:0], [EXE:1], [LMP:1]) .....	175
그림 54	Command-line 방식의 HSBGen.exe ([패킹:0],[실행:0], [EXE:1], [LMP:0]) .....	176
그림 55	Command-line 방식의 HSBGen.exe ([패킹:1],[실행:0], [EXE:0], [LMP:1]) .....	176
그림 56	Command-line 방식의 HSBGen.exe ([패킹:0],[실행:0], [EXE:0], [LMP:1]) .....	176
그림 57	HSBGen.ini.....	179
그림 58	SetServerList.exe 사용 방법 .....	186
그림 59	HSBHelper.exe 사용 방법.....	189

## 표 목차

---

표 1 클라이언트 요구 사항.....	12
표 2 서버(서버 연동) 요구 사항 .....	12
표 3 HackShield 관련 파일 .....	21
표 4 고객 자체 패치 모듈 사용 업데이트 시 필요한 패치 파일 .....	22
표 5 AHNHS_CHKOPT_SELF_DESTRUCTION 적용 시 처리 콜백 코드 .....	35
표 6 HackShield 옵션별 콜백 코드 및 return value .....	38
표 7 업데이트 관련 파일.....	80
표 8 업데이트 PatchSet.....	80
표 9 업데이트 서버에 설치되어야 하는 파일 목록 .....	81
표 10 업데이트에 사용되는 데이터 파일 .....	82
표 11 HackShield 업데이트 이후 자동 생성 파일.....	82
표 12 HackShield 업데이트 서버 사양.....	83
표 13 서버 연동 버전 관리.....	102
표 14 서버 연동 관련 구성 파일 .....	103
표 15 AntiCpXSvr 관련 파일 .....	104
표 16 Specific Error Code .....	116
표 17 모니터링 관련 파일 .....	129
표 18 LMP 관련 파일.....	141
표 19 LMP 지원 패커.....	145
표 20 HsCryptLib 파일 .....	149
표 21 HsUserUtil 파일 .....	161
표 22 HSBGen 툴 시스템 요구 사항 .....	171
표 23 HackShield 기능 OFF .....	194

# **1장**

## **HackShield 소개**

1.1 HackShield 소개 /12  
1.2 시스템 요구 사항 /12

## 1.1 HackShield 소개

HackShield는 10여 년간 축적된 PC 보안 기술을 기반으로 안랩에서 개발한 게임 해킹 툴 탐지, 해킹 차단, 크랙 방지, 실행 파일 실시간 보호, 데이터 암호화 등의 기능을 제공하는 온라인 게임 해킹 방지 솔루션입니다.

HackShield의 대표적인 기능은 다음과 같습니다.

- 해킹 차단 및 해킹툴 탐지 기능: 2 장
- 업데이트: 3 장
- 서버 연동 크랙 방지: 4 장
- 모니터링 서비스: 5 장
- Local Memory Protection(LMP): 6 장
- 데이터 파일/메시지 암호화: 7 장
- User 권한 실행 지원: 8 장

각 기능에 대한 자세한 내용과 프로그래밍 방법은 이 매뉴얼의 각 장을 참고하십시오.

## 1.2 시스템 요구 사항

HackShield의 안정적인 설치와 정상적인 동작을 위해 필요한 클라이언트 및 서버의 시스템 환경은 다음과 같습니다.

표 1 클라이언트 요구 사항

항목	권장 사양	최소 사양
운영체제	Windows 98, ME, 2000, Professional, XP Home, XP Professional, Server 2003, Vista, 7	Windows 98 이상
CPU	Intel Pentium 500Mhz 이상	Intel Pentium 133MHz 이상 IBM-PC 호환기종
RAM	128MB 이상	32MB
하드디스크	2MB 이상	2MB

표 2 서버(서버 연동) 요구 사항

운영체제	플랫폼
Windows 2K, 2K3, XP, VISTA, 2008	x86, x64
Solaris 8, 10 (32bit)	x86
Fedora Linux 7.1(x86), 11(x64), redhat enterprise 4, cent OS 5.3, FreeBSD 7.x(x86)	x86, x64



주의

---

웹 서버, DB 서버 등 서버 구동용으로 사용하는 Windows NT 계열에서 HackShield를 실행할 경우, 예상하지 못한 성능 저하 등의 문제가 발생할 수 있습니다.

---

 참고

CPU는 Intel x86, x64 계열(x86 호환 AMD 계열 포함)만 지원합니다. Windows NT를 실행하는 Alpha chip machine과 Windows를 실행하는 NEC pc 8xxx machine 운영체제는 지원하지 않습니다.

---

# **2장**

# **기본 기능**

**2.1 기본 기능 /15**

**2.2 애플리케이션 프로그래밍 /18**

**2.3 애플리케이션 프로그래밍 인터페이스 /31**

## 2.1 기본 기능

---

HackShield의 해킹 차단 및 해킹툴 탐지 기능을 소개하고 이 기능을 구현하기 위한 방법을 설명합니다.

### 2.1.1 기능 소개

HackShield가 온라인 게임 해킹 방지를 위해 제공하는 기본 기능은 다음과 같습니다.

- 엔진 기반(시그니처/휴리스틱)의 해킹툴 탐지
- 메모리 해킹 차단
- 스피드 핵 차단
- 디버깅 차단
- 오토 마우스 차단

#### 엔진 기반(시그니처/휴리스틱)의 해킹툴 탐지

안랩의 안티 바이러스 기술을 이용하여 해킹툴을 탐지합니다. 엔진에 시그니처가 등록되어 있는 해킹툴인 경우, 필요에 따라 해당 툴을 강제로 종료시키고 해당 프로세스의 이름을 게임 클라이언트에게 콜백 함수로 알립니다.

새로운 해킹툴을 탐지하기 위해 엔진을 최신 버전으로 업데이트하여 해킹툴을 차단합니다.

#### 메모리 해킹 차단

게임에서 사용하는 Windows API(OpenProcess, Read/WriteProcessMemory...)를 사용해 메모리 접근을 차단합니다. 결과 값을 조작하는 등의 해킹을 커널 레벨에서 직접 메모리를 추적하여 차단합니다.



해킹툴이 아니어도 메모리에 직접 접근하는 프로그램의 경우 정상 동작하지 않을 수 있습니다.

---

#### 스피드 핵 차단

스피드 핵은 시스템에 장착되어 있는 타이머를 조작하는 하드웨어 방식과 운영체제의 시간 관련 API를 조작하는 소프트웨어 방식이 있습니다. 이 스피드 핵을 차단하기 위해서 시스템의 시간과 운영체제에서 만들어내는 논리적인 시간의 차이를 마이크로 프로세서 레벨에서 수시로 확인합니다. 시스템의 시간과 운영체제의 시간이 일정 시간 이상 차이가

나면 이를 게임 클라이언트에게 콜백 함수로 알립니다.

사용자의 시스템이나 운영체제, 그리고 게임의 성격에 따라 스피드 핵 감지에 대한 정도의 차이가 있으므로 별도의 파라미터를 통해서 스피드 핵 감지율에 대한 레벨을 5단계로 설정할 수 있습니다.

---

### 참고

스피드 핵이란 타이머 처리 마이크로 프로세서나 Windows 시스템에서 제공하는 시간 관련 함수를 조작하여 시간 관련 기능이 비정상적으로 동작되게 하는 프로그램입니다.

---

## 디버깅 차단

디버거를 이용하여 게임을 분석하거나 해킹을 시도할 수 없도록 모든 디버거의 디버거 트레이싱을 차단합니다. 디버거 트레이싱 시도가 감지되면 게임 클라이언트에게 콜백 함수로 알립니다. HackShield를 초기화할 때에도 SoftICE와 같은 디버깅 프로그램의 실행 여부를 먼저 확인하여 디버깅 프로그램이 실행되고 있으면 에러를 리턴합니다.

## 오토 마우스 차단

마우스 입력을 자동으로 처리해서 게임을 조작하거나 게임 서버에 과부하를 발생시킬 수 없도록 오토 마우스를 차단합니다.

### 2.1.2 기능 특징

HackShield 기본 기능의 특징은 다음과 같습니다.

## 인터페이스 함수(API) 제공

HackShield의 기능을 편리하게 사용하고 그 실행 결과 값을 받아 볼 수 있도록 인터페이스 DLL을 제공합니다. 고객이 고유 정책에 따라 HackShield의 해킹 차단 기능 중 필요한 기능만 실행하려면 인터페이스 DLL을 사용하십시오.

## 샘플 프로그램 제공

HackShield에서 제공하는 API를 사용하여 구현한 테스트용 게임 클라이언트 프로그램을 제공합니다. 고객은 샘플 프로그램을 참고하여 HackShield의 기능 및 성능을 확인하고 클라이언트 프로그램 개발 시 참고할 수 있습니다.

## HackShield 업데이트 제공

HackShield 업데이트 기능을 사용하여 해킹 차단 및 해킹툴 탐지 엔진 등을 수시로 제공합니다. 게임 클라이언트는 지정된 업데이트 서버를 통하여 최신 해킹 차단 및 해킹툴 탐지 엔진을 다운로드할 수 있습니다. 업데이트 서버는 FTP 및 HTTP를 모두 지원하며, FTP의 경우 anonymous 로그인과 사용자 로그인을 모두 지원합니다. 업데이트할 때 옵션을 선택하면 업데이트 진행 상황을 창에서 확인할 수 있습니다.

### 2.1.3 시스템 구조

HackShield는 독립된 실행 파일 형태가 아닌 SDK를 이용한 라이브러리 형태로 제공합니다. HackShield 전체 구조 및 동작 원리는 다음과 같습니다.

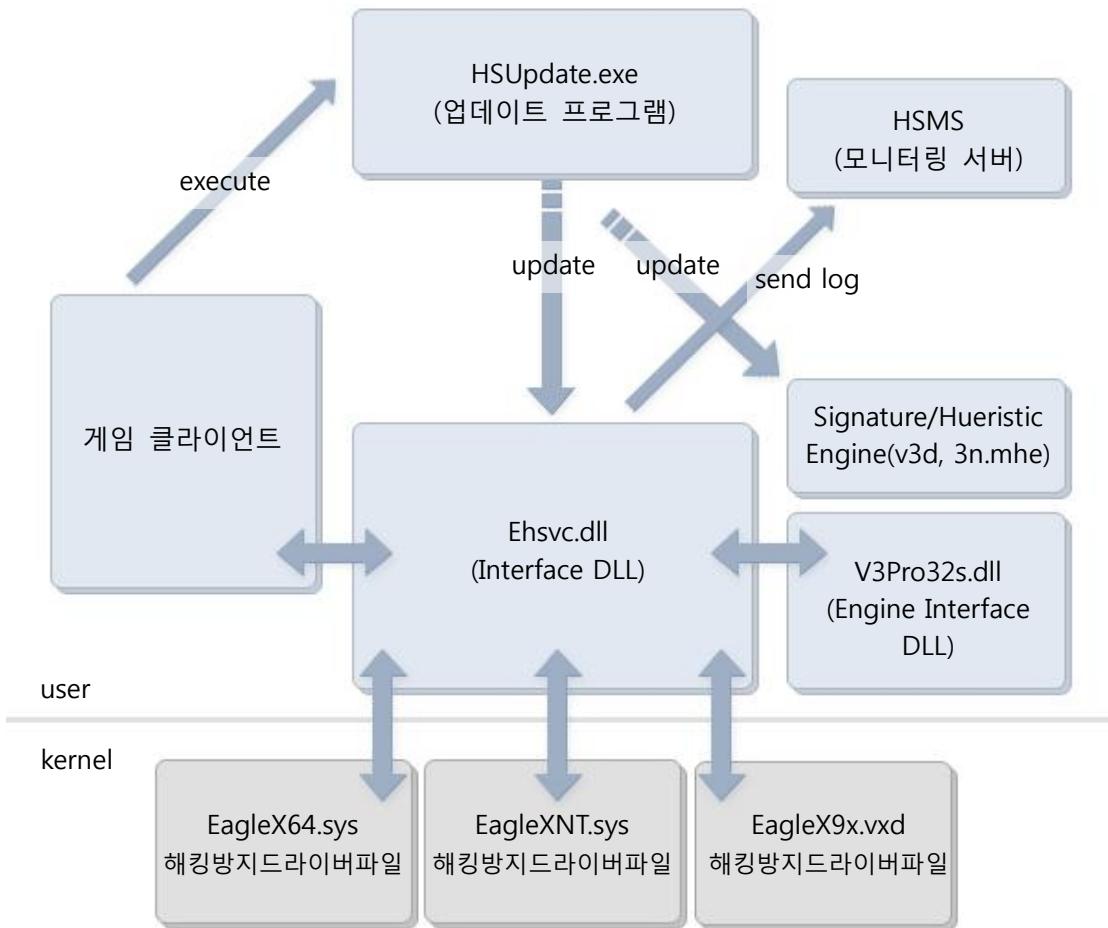


그림 1 HackShield 전체 구조

위 그림의 주요 항목에 대한 설명은 다음과 같습니다.

#### HSUpdate.exe(업데이트 프로그램)

HackShield 전용 업데이트 프로그램입니다. 업데이트 서버에서 해킹 차단 모듈과 해킹툴 탐지 모듈을 포함한 엔진 파일을 다운로드해서 업데이트합니다.

### **EHSvc.dll(인터페이스 DLL)**

해킹 차단 모듈과 해킹툴 탐지 엔진을 동작하게 하는 인터페이스 파일입니다. 해킹툴 탐지 및 해킹 차단 결과를 게임 클라이언트에게 콜백 함수로 알립니다.

### **V3Pro32s.dll(엔진 인터페이스 DLL)**

안랩의 안티 바이러스 기술을 이용한 해킹툴 탐지 엔진을 동작시킵니다. 실행되고 있는 해킹툴의 정보가 엔진 시그니처 파일에 이미 등록되어 있는 경우, 즉시 이를 탐지하여 게임 프로세스에 알립니다. 만약 엔진 시그니처 파일에 등록되어 있지 않은 해킹툴인 경우에는 우선 엔진 시그니처 파일에 등록시켜 즉시 차단하고, 새로운 해킹툴의 동작 방식을 분석한 엔진 시그니처 파일로 엔진을 업데이트합니다.

### **EagleX9x.vxd, EagleXNT.sys or EagleX64.sys(해킹 차단 드라이버)**

커널 모드에서 동작하는 드라이버 파일로서 해킹 차단 역할을 합니다. 인터페이스 DLL인 EHSvc.dll에 포함되어 있으며, 실행되는 운영체제에 따라 EagleX9x.vxd, EagleXNT.sys이나 EagleX64.sys가 시스템에 로드됩니다.

## **2.2 애플리케이션 프로그래밍**

HackShield의 해킹 차단 및 해킹툴 탐지 기능을 구현하는 방법을 설명합니다.



#### **참고**

이 장에서 사용하고 있는 샘플 코드는 Microsoft Visual C++ 6.0을 기준으로 한 C/C++ 언어로 작성했습니다. 프로그래밍에 사용되는 언어는 각 프로그램의 특성 및 시스템 환경에 따라서 변경할 수 있습니다.

### **2.2.1 프로그래밍 순서**

게임 클라이언트 개발자가 해킹툴 탐지 및 해킹 차단 기능을 구현하는 순서는 다음과 같습니다.

1. 함수를 호출하는 코드를 작성하기 전에 HackShield 파일 목록을 확인하고 필요한 파일을 복사합니다.
2. 프로그래밍을 하기 위해 반드시 필요한 라이선스 키를 발급받습니다.
3. HackShield 업데이트를 사용하는 경우, HackShield 모듈이 자동으로 업데이트 될 수 있도록 HackShield 업데이트 함수 AhnHS\_HSUpdateEx()를 작성합니다.

---

 참고

HackShield의 업데이트 프로그램을 이용하여 업데이트를 하는 경우에는 별도로 제공되는 업데이트 라이브러리(HSUpChk.lib)를 사용해서 업데이트 함수를 호출합니다.

---

4. HackShield 모니터링 함수 AhnHS\_StartMonitor()를 작성합니다. 이 함수는 HackShield 모니터링 서버 HSMS 가 설치된 경우, 서버에 해킹 및 에러 정보를 전송합니다.
5. 파일 위/변조 검사 및 데이터 초기화를 위한 HackShield 초기화 함수 AhnHS\_Initialize()를 작성합니다.
6. 해킹 차단 및 해킹툴을 탐지하기 위해 HackShield 시작 함수 AhnHS\_StartService()를 작성합니다.
7. 해킹 차단 기능 및 해킹툴 탐지 기능 실행 결과를 처리하기 위한 HackShield 콜백 함수 AhnHS\_Callback()를 작성합니다.
8. 해킹 차단 기능 및 해킹툴 탐지 기능을 정지시키기 위해 프로그램 종료 처리 부분에 HackShield 정지 함수 AhnHS\_StopService()를 작성합니다.
9. HackShield 정지 함수를 호출한 다음 HackShield 완료 함수 AhnHS\_Uninitialize()를 작성합니다.

---

 주의

게임이 비정상적으로 종료된 경우에도 HackShield의 종료 루틴은 반드시 실행되어야 합니다. 비정상적인 종료 상황에 대비하여 게임의 최초 시작 위치에 다음의 코드를 추가하십시오. 예외로 인한 게임 종료 시에도 HackShield 종료 루틴을 실행할 수 있습니다.

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

---

10. 작성한 소스 코드가 정상적으로 동작하는지 테스트합니다.

11. 게임 클라이언트를 게임 유저에게 배포합니다.

지금까지의 과정을 순서대로 간단히 나타내면 다음과 같습니다.

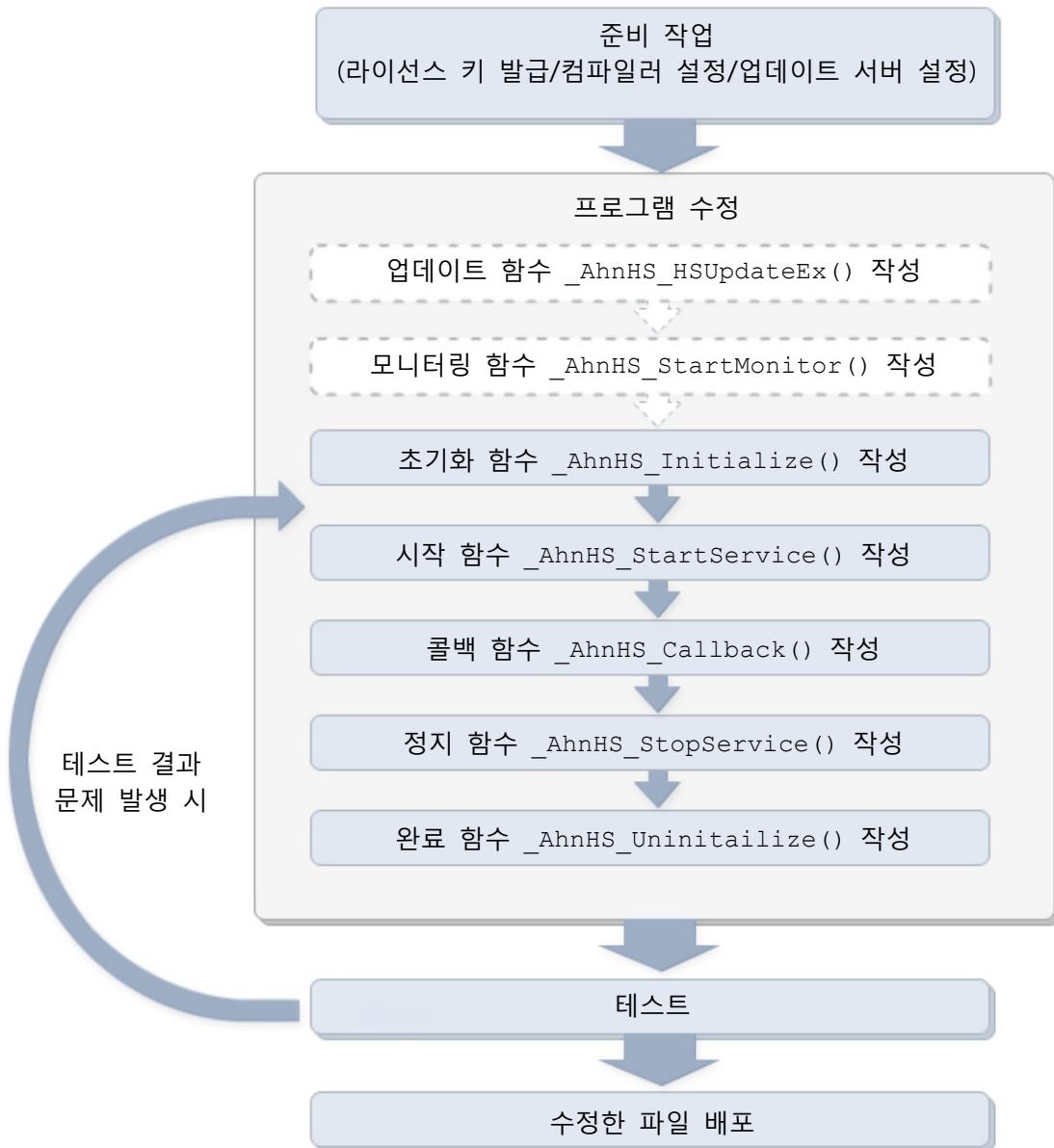


그림 2 애플리케이션 프로그래밍 순서

### 2.2.2 프로그래밍 준비

HackShield를 사용하여 프로그래밍을 시작하기 전에 설치 디렉터리에서 확인해야 하는 HackShield 파일 목록과 컴파일러 설정 방법, 업데이트 모듈 사용 방법, 업데이트 서버 설정 방법, 라이선스 키 발급 방법은 다음과 같습니다.

#### 설치 디렉터리

고객은 게임 클라이언트가 설치되어 있는 폴더 하위에 다음과 같이 HackShield 폴더를 만들어서 HackShield 관련 파일들을 설치합니다.

<Game Directory>\WHShield\

## HackShield 관련 파일

<Game Directory>\WHShield\에 설치해야 할 HackShield 관련 파일은 다음과 같습니다.

표 3 HackShield 관련 파일

파일 이름	설치 폴더	설명
3n.mhe	[HackShield 폴더]	휴리스틱 엔진 파일
EhSvc.dll	[HackShield 폴더]	인터페이스 DLL
hshield.dat	[HackShield 폴더]	HackShield 관련 dat 파일
v3pro32s.dll	[HackShield 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
asc_com.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
asc_dh.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
asc_fse.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
asc_intg.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
asc_mmgr.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
asc_unp.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
fse_base.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
fse_fact.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
fse_pe.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
gfs_base.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
gfs_fact.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
gfs_file.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
gfs_mem.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
gfs_os.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
gfs_proc.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
gfs_util.dll	[asc 폴더]	해킹툴 탐지 엔진 인터페이스 DLL
Oasc.scd	[asc 폴더]	해킹 차단 엔진 패턴 파일
Oscsecure.scd	[asc 폴더]	해킹 차단 엔진 패턴 파일
Osgame.scd	[asc 폴더]	해킹 차단 엔진 패턴 파일
Ospe3f.scd	[asc 폴더]	해킹 차단 엔진 패턴 파일
moduler.scd	[asc 폴더]	해킹 차단 엔진 패턴 파일
option.scd	[asc 폴더]	해킹 차단 엔진 패턴 파일
AhnRpt.exe	[HackShield 폴더]	HackShield 로그 수집 파일
HsLogMgr.exe	[HackShield 폴더]	HackShield 로그 수집 선택 파일
AhnRpt.ini	[HackShield 폴더]	HackShield 로그 수집 정보 파일
HShield.h	[게임 소스 폴더]	헤더 파일
HShield.lib	[게임 소스 폴더]	라이브러리 파일

## 컴파일러 설정

HackShield를 사용하는 게임 클라이언트 프로그램 프로젝트 파일의 라이브러리나 소스 코드 목록에 HShield.lib를 포함시켜 컴파일러를 설정합니다.

### 2.2.3 프로그래밍 적용

프로그래밍을 하기 위해 필요한 라이선스 키 발급 방법과 HackShield 기본 함수의 코드작성 방법은 다음과 같습니다.

#### HackShield 라이선스 키 발급

HackShield를 적용하려면 반드시 안랩으로부터 부여받은 라이선스 키가 필요합니다. 라이선스 키를 발급받는 방법은 다음과 같습니다.

1. EhSvc.dll 을 사용하는 실행 파일의 이름, 고객의 지역, 고객 이름, 게임 이름을 안랩 HackShield 기술 지원에 전달하여 라이선스 키 발급을 요청합니다.
2. 고유의 게임 코드인 4 자리 숫자와 24 자리 문자열로 구성된 라이선스 키가 발급됩니다.
3. 발급 받은 게임 코드와 라이선스 키 값을 초기화 함수 \_AhnHS\_Initialize()의 파라미터로 전달합니다.

---

#### ⚠ 주의

초기화 함수 \_AhnHS\_Initialize()의 파라미터로 잘못된 값을 전달하면 초기화 함수가 정상적으로 호출되지 못하고 에러(HS\_ERR\_INVALID\_LICENSE)를 리턴합니다.

---

#### HackShield 업데이트 함수 작성

HackShield 업데이트 함수를 작성하는 방법은 HackShield 업데이트 모듈을 사용하는 방법과 고객 자체 패치 모듈을 사용하는 방법이 있습니다.

- HackShield 업데이트 모듈을 사용하는 방법
  - 업데이트 서버 설정 및 클라이언트 적용()

---

#### 💡 참고

업데이트 서버 설정 및 클라이언트 적용에 대한 자세한 내용은 [3장 HackShield 업데이트 기능](#)을 참고하십시오.

---

- 클라이언트 적용 시 HackShield 의 인터페이스 DLL 인 EhSvc.dll 파일이 로드되기 전에 업데이트 라이브러리 함수를 호출해서 HackShield 업데이트를 실행
- 고객 자체 패치 모듈(업데이트 서버) 사용
  - 별도의 업데이트 모듈을 배포하지 않고 기존의 게임 패치 모듈과 함께 HackShield 관련 파일들만 포함시켜 배포

고객 자체 패치 모듈을 사용하여 업데이트하는 경우, HackShield 관련 패치 파일은 다음과 같습니다.

**표 4 고객 자체 패치 모듈 사용 업데이트 시 필요한 패치 파일**

파일 이름	설명
3n.mhe	휴리스틱 엔진 파일

Ehsvc.dll	HackShield 인터페이스 DLL
hshield.dat	HackShield 관련 dat 파일
V3pro32s.dll	해킹툴 탐지 엔진 인터페이스 DLL
Oasc.scd	해킹툴 패턴 엔진 파일
Osccure.scd	해킹툴 패턴 엔진 파일
Osgame.scd	해킹툴 패턴 엔진 파일
Ospe3f.scd	해킹툴 패턴 엔진 파일
asc_com.dll	해킹툴 탐지 엔진 인터페이스 DLL
asc_dh.dll	해킹툴 탐지 엔진 인터페이스 DLL
asc_fse.dll	해킹툴 탐지 엔진 인터페이스 DLL
asc_intg.dll	해킹툴 탐지 엔진 인터페이스 DLL
asc_mmgr.dll	해킹툴 탐지 엔진 인터페이스 DLL
asc_unp.dll	해킹툴 탐지 엔진 인터페이스 DLL
fse_base.dll	해킹툴 탐지 엔진 인터페이스 DLL
fse_fact.dll	해킹툴 탐지 엔진 인터페이스 DLL
fse_pe.dll	해킹툴 탐지 엔진 인터페이스 DLL
gfs_base.dll	해킹툴 탐지 엔진 인터페이스 DLL
gfs_fact.dll	해킹툴 탐지 엔진 인터페이스 DLL
gfs_file.dll	해킹툴 탐지 엔진 인터페이스 DLL
gfs_mem.dll	해킹툴 탐지 엔진 인터페이스 DLL
gfs_os.dll	해킹툴 탐지 엔진 인터페이스 DLL
gfs_proc.dll	해킹툴 탐지 엔진 인터페이스 DLL
gfs_util.dll	해킹툴 탐지 엔진 인터페이스 DLL
moduler.scd	해킹툴 패턴 엔진 파일
option.scd	해킹툴 패턴 엔진 파일

## HackShield 모니터링 함수 작성

모니터링 서버의 설치 및 운영 방법은 AhnLab HSMS Install Guide와 AhnLab HSMS Operator Guide를 참고하십시오.



참고

클라이언트 부분은 [5장 모니터링 서비스 기능](#)을 참고하여 모니터링 기능을 적용하십시오.

## HackShield 초기화 함수 작성

프로그래밍을 위한 준비 작업이 끝났으면 가장 먼저 초기화 함수 `_AhnHS_Initialize()`를 호출합니다. 초기화 함수 호출이 성공적으로 완료되어야 HackShield의 해킹 차단 기능 및 해킹툴 탐지 기능을 실행할 수 있습니다.

초기화 함수는 게임 클라이언트 프로그램의 인스턴스를 초기화하거나 메인 윈도우를 초기화하는 루틴에서 호출합니다.

`_AhnHS_Initialize()` 호출 예제는 다음과 같습니다.

```
BOOL HS_Init()
```

```

{
    int nRet = 0;
    TCHAR szFullPath[MAX_PATH];
    TCHAR szMsg[MAX_PATH];
    DWORD dwOption = 0;

    // ① HackShield 폴더의 EhSvc.dll 위치를 지정합니다.
    lstrcat ( szFullPath, _T( "\\\HShield\\EhSvc.dll" ) );

    // ② _AhnHS_Initialize() 호출에 쓰일 옵션 플래그를 정의합니다
    dwOption = AHNHS_CHKOPT_ALL;

    // ③ _AhnHS_Initialize()를 호출하여 HackShield를 초기화합니다.

    nRet = _AhnHS_Initialize (
        szFullPath,
        HS_CallbackProc,           // 콜백 함수
        1000,                      // 게임 코드
        "B228F291B7D7FAD361D7A4B7", // 라이선스 키
        dwOption,                  // 옵션 플래그
        AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL);

    // AHNHS_CHKOPT_SELF_DESTRUCTION 기능 사용 시 자체 종료 대기 시간을
    // 조정 하시려면 아래와 같이 적용 하시면 됩니다.
    //uint unAdditionalRatio = AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL
    //                           | AHNHS_SELFDESTRUCTION_RATIO_NORMAL;
    //nRet = _AhnHS_Initialize (
    //        szFullPath,
    //        HS_CallbackProc,           // 콜백 함수
    //        1000,                      // 게임 코드
    //        "B228F291B7D7FAD361D7A4B7", // 라이선스 키
    //        dwOption,                  // 옵션 플래그
    //        unAdditionalRatio );

    // ④ _AhnHS_Initialize()의 반환값을 검사하여 에러 처리합니다.
    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, "해킹방지 기능에 문제가 발생하였습니다. (%x)" , nRet );
        break;
    }
    MessageBox( NULL, szMsg, szTitle, MB_OK );
    return FALSE;
}
return TRUE;
}

```

**그림 3 \_AhnHS\_Initialize() 호출 예제**

위 예제에서는 모든 옵션을 사용하도록 초기화 옵션을 설정하는 dwOption에 AHNHS\_CHKOPT\_ALL을 사용했습니다. 단, 옵션 중에는 선택하지 않아도 내부적으로 자동 적용되는 코어 옵션이 있습니다.

초기화 함수의 마지막 파라미터는 스피드 핵 감지율 레벨과 핵쉴드 자체 종료 대기 시간을 의미합니다. 스피드 핵 감지율 레벨은 단지 SPEEDHACK\_SENSING\_RATIO만을 사용할 수도 있지만 게임의 속도를 느리게 만드는 행위가 유리하게 작용하지 않는 게임의 경우, 추가 옵션인 AHNHS\_SPEEDHACK\_EXOPT\_ALLOW\_SLOW를 함께 사용할 수 있습니다. 핵쉴드 자체 종료 대기 시간은 AHNHS\_CHKOPT\_SELF\_DESTRUCTION옵션이 적용되어 있을 때 동작합니다. AHNHS\_SELFDESTRUCTION\_RATIO를 적용하지 않으면 기본적으로 AHNHS\_SELFDESTRUCTION\_RATIO\_NORMAL이 적용됩니다.

#### ⚠ 주의

초기화 함수 \_AhnHS\_Initialize()는 한 프로세스당 한 번만 호출할 수 있습니다. 초기화 함수를 중복 호출하면 에러가 발생할 수 있습니다.

EhSvc.dll 파일이 위/변조됐거나 버전이 맞지 않을 경우에는 초기화 함수를 호출하면 에러 값(HS\_ERR\_INVALID\_FILES)을 리턴합니다.

#### ➊ 참고

HackShield 인터페이스 DLL 파일 EhSvc.dll이 아닌 게임 클라이언트 프로그램에서 HackShield 함수를 호출하는 부분이 위/변조될 가능성도 있습니다. 따라서 게임 클라이언트 프로그램에서도 파일 위/변조를 차단할 수 있도록 Packer 프로그램을 사용하여 게임 클라이언트 파일을 암호화, 압축하여 배포할 것을 권장합니다.

그러나 HackShield에서는 서버 연동 클라이언트 크랙 감시 솔루션을 이용해서 크랙한 실행 파일로 게임을 실행하는 것을 원천 차단할 수 있는 기능을 제공합니다.

일부 사용자나 해커는 Windows XP에서 제공하는 하위 호환성 모드인 Compatibility Mode로 프로그램을 실행함으로써 프로그램이 오동작하도록 시도합니다. 하위 호환성 모드로 실행될 경우 현재 시스템이 Windows XP이더라도 Windows 98이나 ME, Windows 2000 등의 운영체제인 것처럼 프로그램이 판단하게 되므로 예기치 못한 결과를 초래할 수 있습니다. 만약 호환성 모드에서 프로그램이 실행 중인 경우 HackShield 모듈의 초기화 함수를 호출하면 에러 값 HS\_ERR\_COMPATIBILITY\_MODE\_RUNNING을 리턴합니다.

초기화 함수를 호출할 때 기본 기능 옵션 이외에 부가 기능 옵션을 선택할 수 있습니다.

## HackShield 시작 함수 작성

HackShield의 해킹 차단 및 해킹툴 탐지 기능을 실행하기 위해서는 시작 함수 `_AhnHS_StartService()`를 호출합니다. `_AhnHS_StartService()` 호출 예제는 다음과 같습니다.

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

    // ① _AhnHS_StartService()를 호출하여 HackShield를 시작합니다.
    nRet = _AhnHS_StartService();

    // ② _AhnHS_StartService()의 반환값을 검사하여 에러 처리합니다.
    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, "해킹 방지 기능에 문제가 발생 하였습니다. (%x)", nRet );
                break;
        }

        MessageBox( NULL, szMsg, szTitle, MB_OK );
        return FALSE;
    }
    return TRUE;
}
```

그림 4 `_AhnHS_StartService()` 호출 예제

HackShield 시작 함수 `_AhnHS_StartService()`는 초기화 함수 `_AhnHS_Initialize()`가 정상적으로 호출 완료된 다음 호출해야 합니다.

---

### ⚠ 주의

게임 클라이언트 프로그램을 좀 더 철저히 보호하기 위해서는 초기화 함수 호출과 거의 동시에 HackShield 시작 함수를 호출하는 것이 좋습니다. 시작 함수를 늦게 호출할수록 해킹툴이 게임 클라이언트에 침입할 가능성이 높아집니다.

## HackShield 콜백 함수 작성

HackShield 시작 함수를 호출하면, 현재 상태를 실시간으로 이벤트로 알립니다. 콜백 함수 호출 예제는 다음과 같습니다.

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

    // ① 각 경우에 대하여 알맞은 에러 메시지를 출력합니다
    switch ( lCode )
    {
        // ② Engine 콜백
        case AHNHS_ENGINE_DETECT_GAME_HACK:
            wsprintf( szMsg, "다음의 프로그램과 게임이 함께 실행될 수 없습니다.
(%x) \n [%s]" , lCode, (LPTSTR)pParam );
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;

        // ③ 오토매크로 탐지
        case AHNHS_ACTAPC_DETECT_AUTOMACRO:
            wsprintf(szMsg, _T("매크로 기능으로 의심되는 동작이 감지되었습니다.
(Code = %x)" , lCode));
            MessageBox(NULL, szMsg, szTitle, MB_OK);
            break;

        // ④ Speedhack 관련
        case AHNHS_ACTAPC_DETECT_SPEEDHACK:
            wsprintf( szMsg, "스피드 핵으로 의심되는 동작이 감지되었습니다.
(%x)" , lCode );
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;

        // ⑤ 디버깅 방지
        case AHNHS_ACTAPC_DETECT_KDTRACE:
        case AHNHS_ACTAPC_DETECT_KDTRACE_CHANGED:
            wsprintf( szMsg, "게임에 대하여 디버깅 시도가 감지되었습니다.
(%x)" , lCode );
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;

        // ⑥ HackShield가 정상 동작 중임
        case AHNHS_ACTAPC_STATUS_HACKSHIELD_RUNNING:
            // HackShield가 정상적으로 동작하고 있는 상태입니다.
            // 고객이 정의한 로직을 실행하도록 합니다.
            DWORD *dwParam = (DWORD*)pParam;
            break;

        // ⑦ 그 외 해킹 방지 기능 이상
        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_ACTAPC_DETECT_RMEM MODIFY_FROM_LMP:
        case AHNHS_AHNHS_ACTAPC_DETECT_ENGINERUNNING:
        case AHNHS_ACTAPC_DETECT_CODEMISMATCH:
        case AHNHS_ACTAPC_DETECT_ANTIFREESERVER:
        case AHNHS_ACTAPC_DETECT_ABNORMAL_HACKSHIELD_STATUS:
            wsprintf( szMsg, “해킹 방어 기능에 이상이 발생하였습니다. (%x)”, lCode );
            MessageBox( NULL, szMsg, szTitle, MB_OK );
            break;
        return 1;
    }

```

**그림 5 콜백 함수 호출 예제**

#### ⚠ 주의

AHNHS\_ACTAPC\_STATUS\_HACKSHIELD\_RUNNING Callback의 경우 HackShield 동작 유무를 확인하는 \_AhnHS\_CheckHackShieldRunningStatus()를 호출하는 경우에 발생하는 콜백입니다. 이 콜백은 HackShield가 동작 중인 경우에는 주기적으로 발생하며, 주기적으로 콜백 처리가 되지 않는다면 HackShield의 이상 동작을 의심할 수 있습니다. 콜백 발생 이후 로직은 고객이 정책에 맞게 구현합니다.

#### ⚠ 주의

고객이 유니코드를 지원할 경우, param을 변환하여 사용하는 방법은 다음과 같습니다.

```

case AHNHS_ENGINE_DETECT_GAME_HACK:
{
    WCHAR wszPath[MAX_PATH] = {0, };
    MultiByteToWideChar
        (CP_APC, 0, (LPVOID)pParam, lstrlen(pParam), wszPath, MAX_PATH)
    wsprintf( szMsg,
        “다음의 프로그램과 게임이 함께 실행될 수 없습니다. (%x) \n [%s]”,
        lCode, (LPTSTR)pParam );
    MessageBox( NULL, szMsg, szTitle, MB_OK );
    break;
}

```

## HackShield 정지 함수 작성

게임 클라이언트 프로그램 종료 전에 먼저 HackShield 정지 함수 \_AhnHS\_StopService()를 호출하여 해킹 차단 기능 및 해킹툴 탐지 기능을 정지합니다. 게임 클라이언트 프로그램을 종료하지 않고 HackShield만 잠시 중지하려면 정지 함수 \_AhnHS\_StopService()를 호출한 다음에 시작 함수 \_AhnHS\_StartService()를 다시 호출합니다.

\_AhnHS\_StopService() 호출 예제는 다음과 같습니다.

```

BOOL HS_StopService()
{
    int nRet = 0;

    // ① _AhnHS_StopService()를 호출하여 HackShield를 정지합니다.
    nRet = _AhnHS_StopService();

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

```

그림 6 \_AhnHS\_StopService() 호출 예제

## HackShield 종료 함수 작성

HackShield 종료 함수 \_AhnHS\_Uninitialize()는 게임 클라이언트 프로그램에 사용된 메모리를 해제시켜 해당 프로그램이 완전히 종료되게 합니다.

\_AhnHS\_Uninitialize() 호출 예제는 다음과 같습니다.

```

BOOL HS_UnInit()
{
    int nRet = 0;

    // ① _AhnHS_Uninitialize()를 호출하여 HackShield를 종료합니다.
    nRet = _AhnHS_Uninitialize();

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

```

그림 7 \_AhnHS\_Uninitialize() 호출 예제

HackShield 종료 함수 \_AhnHS\_Uninitialize()가 호출되고 나면 HackShield는 더 이상 게임 클라이언트를 보호하지 않습니다. 따라서 \_AhnHS\_Uninitialize()는 게임 클라이언트가 종료되는 마지막 시점에 호출해야 합니다.

### ⚠ 주의

게임 클라이언트 프로그램이 비정상적으로 종료되면 HackShield의 해킹 차단 드라이버를 언로드하지 못할 수 있습니다. 만약 이 때 초기화 함수 \_AhnHS\_Initialize()를 다시 호출하면, 사용자 시스템에 HackShield의 해킹 차단 드라이버가 이미 로드되어 있기 때문에 에러 HS\_ERR\_INIT\_DRV\_FAILED가 리턴됩니다. 따라서 이러한 경우에는 시스템을 재부팅한 다음 게임 클라이언트 프로그램

---

을 처음부터 다시 실행하십시오.

---

## HackShield 로그 수집 함수 작성

HackShield 로그 수집 함수 `_AhnHS_SendHsLog()`를 호출하면 HackShield와 관련된 여러 가지 문제를 고객을 거치지 않고 빠르게 처리할 수 있습니다. 처리하는 순서는 다음과 같습니다.

1. 해킹이 발생한 후 HackShield 로그가 생성됩니다.
2. 콜백 코드를 처리하는 메시지 박스가 나타나면 **HackShield 로그 전송**을 누릅니다.
3. HackShield 콜백 코드에서 받은 이벤트 코드(에러 코드), 게임 유저 아이디, HackShield 경로 값이 파라미터 값으로 들어간 `_AhnHS_SendHsLog()`가 호출되면서 AhnReport 실행 창이 나타납니다.
4. 게임 유저는 AhnReport 실행 창에 에러상황 및 본인의 이메일 주소를 작성한 다음 **AhnReport 실행**을 누릅니다.
5. 게임 유저가 입력한 정보와 함께 HackShield 로그가 안랩의 로그 수집 서버로 전송됩니다.

지금까지의 과정을 그림으로 나타내면 다음과 같습니다.

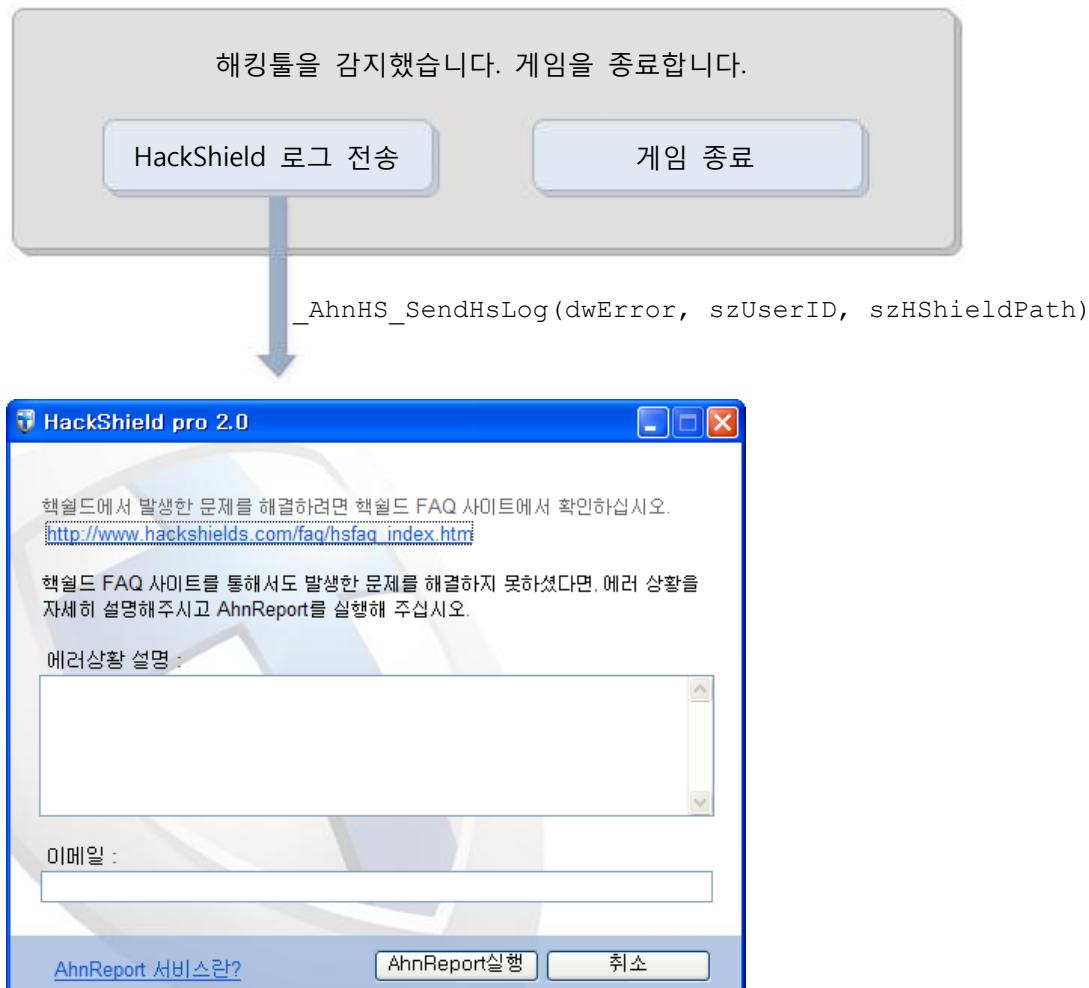


그림 8 HackShield 로그 전송

---

### **⚠ 주의**

게임 유저가 선택하면 AhnReport 실행 창이 나타나는 그림 8의 처리 과정은 반드시 HackShield 에러로 게임이 종료되는 경우에만 사용하여 개발하십시오. HackShield 에러가 아닌 다른 문제로 게임이 종료되는 경우에는 AhnReport 실행 창이 나타나지 않는ダイ얼로그/메시지 박스를 사용해야 합니다.

---

### **⚠ 주의**

\_AhnHS\_SendHsLog ()의 파라미터 szHShieldPath는 Ehsvc.dll의 경로 값이기 때문에 Ehsvc.dll을 제외한 폴더 경로만 작성하십시오.

예제) 경로가 %GameRoot%\\HShield인 경우

```
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');
_sprintf(szHShieldPath, _T("%s\\HShield"), szPath);
_AhnHS_SendHsLog(dwError, "UserID", szHShieldPath);
```

---

### **➊ 참고**

AhnReport 실행 창에서 에러상황 설명 및 이메일은 옵션이므로 작성하지 않아도 됩니다. 전송된 HackShield 로그는 오탐 및 에러상황을 참고하기 위한 로그 정보이기 때문에 모든 고객에게 반드시 답변을 전송해야 하는 것이 아닙니다.

---

### **➊ 참고**

AhnReport 실행 창은 Windows 2000, xp, 2003, vista, 7을 지원하며 win9x를 지원하지 않습니다. 언어는 한국어와 영어, 중국어(간체/번체), 일본어를 지원합니다.

---

## **2.3 애플리케이션 프로그래밍 인터페이스**

---

HackShield의 해킹 차단 및 해킹툴 탐지 기능 구현에 필요한 API를 소개합니다.

### **2.3.1 \_AhnHS\_Initialize()**

HackShield를 초기화하고 옵션을 설정하는 함수입니다. 프로그램이 초기화될 때 한 번만 호출할 수 있습니다. 다른 게임에서 HackShield를 사용하고 있거나 HackShield가 비정상적으로 종료됐을 경우 에러가 발생할 수 있습니다.

```
EHSVC_API
int __stdcall
```

```

    _AhnHS_Initialize(
        const char* szFileName
        PFN_AhnHS_Callback pfn_Callback,
        int nGameCode,
        const char* szLicenseKey,
        DWORD dwOption,
        UINT unAdditionalRatio
    );

```

## Parameters

Parameter	Value	Description
szFileName		EHSvc.dll의 전체 경로
pfn_Callback		콜백 함수의 포인터
nGameCode	4자리 숫자	각 게임에 해당하는 고유 ID 코드
szLicenseKe	24자리 문자열	각 게임에 해당하는 라이선스 키
dwOption		초기화 옵션 설정
unAdditionalRatio		스피드 핵 감지율 레벨(필수) 자체 종료 대기 시간 레벨(선택)

## Option

### AHNHS\_CHKOPT\_SPEEDHACK (코어 옵션)

스피드 핵 차단 기능을 사용합니다. 하드웨어 제어나 소프트웨어 방식의 스피드 핵이 발견되면 스피드 핵 동작 여부를 콜백 함수로 알립니다. 스피드 핵을 진단하는 감지율 레벨 설정에 따라 감지 민감도가 달라집니다. SPEEDHACK SENSING RATIO를 참고하십시오.

#### 참고

코어 옵션이란 함수 호출 시 옵션으로 적용하지 않아도 자동으로 적용되는 옵션을 말합니다.

### AHNHS\_CHKOPT\_READWRITEPROCESSMEMORY (코어 옵션)

HackShield DLL을 사용하고 있는 현재 프로세스의 메모리를 외부 프로세스가 읽기/쓰기를 할 수 없도록 차단합니다.

### AHNHS\_CHKOPT\_KDTRACER (코어 옵션)

커널 모드 디버거 실행을 감지하여 게임 프로세스에 알리는 옵션입니다.

### AHNHS\_CHKOPT\_OPENPROCESS (코어 옵션)

HackShield DLL을 사용하고 있는 현재 프로세스의 정보를 얻을 수 있는 OpenProcess API 함수의 호출을 차단하는 옵션입니다.

#### **AHNHS\_CHKOPT\_AUTOMOUSE (코어 옵션)**

오토 마우스 종류의 프로그램이 현재 프로세스에 아무런 영향을 미칠 수 없도록 차단하는 옵션입니다.

#### **AHNHS\_CHKOPT\_PROCESSSCAN (코어 옵션)**

주기적으로 프로세스 목록을 검사하여 해킹툴 실행 여부를 검사하는 옵션입니다.

#### **AHNHS\_CHKOPT\_ALL**

위에서 설명한 옵션을 모두 포함하는 옵션입니다.

#### **AHNHS\_CHKOPT\_MESSAGEHOOK**

메시지 후킹을 차단하는 옵션입니다.

#### **AHNHS\_CHKOPT\_LOCAL\_MEMORY\_PROTECTION**

지정한 보호 대상 파일의 메모리 영역을 보호하는 옵션입니다.

#### **AHNHS\_CHKOPT\_ANTIFREESERVER**

게임 서버가 아닌 다른 주소로 접속하는 것을 방지하는 옵션입니다.

#### **AHNHS\_CHKOPT\_SEND\_MONITOR\_DELAY**

게임 유저가 ID를 입력하기 전에 해킹툴이 감지됐을 경우, 게임 유저 ID가 입력될 때까지 일정 시간 대기한 다음 모니터링 서버로 게임 유저 정보와 해킹툴 감지 로그를 전송하는 옵션입니다.

#### **AHNHS\_USE\_LOG\_FILE**

HackShield 동작과 관련된 로그를 기록하는 옵션입니다. 이 로그는 문제 발생 시 원인 분석에 사용합니다. 로그 파일은 암호화되어 있기 때문에 별도의 프로그램을 통해서만 내용을 확인할 수 있습니다. 로그 파일은 EhSvc.dll이 위치한 디렉터리에 hshield.log라는 이름으로 만들어집니다.

#### **AHNHS\_ALLOW\_SVCHOST\_OPENPROCESS**

NT 계열의 운영체제에서 구동되는 서비스 프로그램인 SvcHost에서 게임 프로세스를 오픈 할 수 있게 하는 옵션입니다. 클라이언트 프로세스 오픈 및 사운드 출력은 Windows XP의 새로운 서비스인 Windows Audio를 사용합니다. OpenProcess 차단 옵션을 사용하면 사운드가 정상적으로 출력되지 않을 수 있습니다. 따라서 Windows XP에서 DirectMusic 또는 DirectSound로 사운드가 정상적으로 출력되지 않는 경우, 이 옵션을 사용해야 합니다.

#### **AHNHS\_ALLOW\_LSASS\_OPENPROCESS**

NT 계열의 운영체제에서 구동되는 서비스 프로그램인 LSASS.exe에서 게임 프로세스를 오픈할 수 있게 하는 옵션입니다. 클라이언트 내부에서 직접 신용카드 결재 컨트롤을 포함하는 경우, LSASS.exe에서 게임 프로세스에 접근하게 되므로 프로세스 오픈을 허용하기 위해 이 옵션을 사용해야 합니다.

#### AHNHS\_ALLOW\_CSRSS\_OPENPROCESS

NT 계열의 운영체제에서 구동되는 서비스 프로그램인 CSRSS.exe에서 게임 프로세스를 오픈할 수 있게 하는 옵션입니다. 클라이언트 내부에서 직접 신용카드 결재 컨트롤을 포함하는 경우, CSRSS.exe에서 게임 프로세스에 접근하게 되므로 프로세스 오픈을 허용하기 위해 이 옵션을 사용해야 합니다.

#### AHNHS\_DONOT\_TERMINATE\_PROCESS

프로세스 목록을 검사하여 해킹툴이 탐지됐을 때 해당 해킹툴 프로세스를 강제 종료시키는 것이 기본값이지만 해킹툴 프로세스를 강제 종료시키지 않고자 하는 경우 사용하는 옵션입니다. AHNHS\_CHKOPT\_PROCESSSCAN 옵션을 사용할 경우 이 옵션과 같이 사용할 것을 권장합니다.

#### AHNHS\_DISPLAY\_HACKSHIELD\_LOGO

초기화 시 HackShield가 동작하고 있다는 것을 보여주기 위해 HackShield 로고를 화면에 띄우는 옵션입니다. 별도의 스레드로 동작하며 로고는 2초 후 자동으로 사라집니다.

---

##### 참고

고객이 제공하는 HackShield 로고 이미지를 띄우려면 사용할 이미지 파일 이름을 hslogo.jpg로 변경한 다음 [Game Directory]\WHShield 폴더에 저장하십시오.

---

#### AHNHS\_CHKOPT\_PROTECTSCREEN

게임 내에서 스크린 정보를 얻어가는 유형의 해킹을 좀 더 효과적으로 방지할 수 있는 옵션입니다.

---

##### 주의

이 옵션을 사용하면 다른 프로그램의 사용자 인터페이스가 손상되거나 화면캡처 프로그램이 정상적으로 동작하지 않을 수 있습니다. 반드시 안랩 HackShield 기술 지원에 문의한 다음 옵션을 사용하십시오.

---

#### AHNHS\_ALLOW\_SWITCH\_WINDOW

전체 화면 모드로 실행되는 게임을 강제로 창 모드로 실행시키는 해킹툴을 탐지하고, 게임 중 다른 애플리케이션과 화면 전환을 가능하게 하는 옵션입니다. 창 모드로 실행이 불 가능한 게임에 대해서만 적용되는 사항이므로 전체 화면 모드와 창 모드를 모두 지원하는 게임에서는 사용할 수 없습니다.

### AHNHS\_CHKOPT\_STANDALONE

HackShield의 단독 실행 옵션입니다. HackShield는 기본적으로 멀티 로딩을 지원하지만, 이 옵션을 사용하면 동일한 게임 코드를 사용하는 게임에서 중복 실행이 불가능하도록 HackShield에서 에러 코드를 리턴합니다.

### AHNHS\_CHKOPT\_PROTECT\_D3DX

DirectX 모듈의 VTable을 후킹하는 해킹툴 차단 기능을 활성화하는 옵션입니다. 이 옵션은 Windows NT 이상의 운영체제에서만 동작합니다. x64는 지원하지 않습니다.

---

#### !

#### 참고

FPS 게임은 해킹툴의 90% 이상이 DirectX 관련 툴이므로 해당 옵션 사용을 권장합니다.

---

### AHNHS\_CHKOPT\_SELF\_DESTRUCTION

HackShield에서 해킹 행위를 감지하여 콜백 함수를 호출했는데도 게임 프로세스가 종료되지 않은 경우, 일정 시간 후에 HackShield 자체적으로 프로세스를 종료시키는 옵션입니다. 이 옵션을 사용하면 콜백 함수를 수정하여 HackShield 핵 감지 기능을 무력화 시키는 해킹툴에 효과적으로 대응할 수 있습니다.

하지만 HackShield 자체 종료는 비정상적인 종료이기 때문에 해당 옵션 사용 시 반드시 콜백 함수에서 HackShield 에러가 리턴된 이후 자체 종료 대기 시간 이내에 게임 프로세스를 정상적으로 종료해야 합니다. 콜백 함수의 콜백 코드에 대한 적절한 처리를 하고 게임 프로세스를 종료시킬 경우, 참고해야 하는 내용은 다음과 같습니다.

---

#### !

#### 참고

자체 종료 대기 시간은 SELF-DESTRUCTION TIME RATIO의 내용을 참고 하시기 바랍니다.

---

**표 5 AHNHS\_CHKOPT\_SELF\_DESTRUCTION 적용 시 처리 콜백 코드**

적용 옵션	콜백 함수에 적용해야 하는 콜백 코드
필수 적용 콜백 코드	AHNHS_ACTAPC_DETECT_AUTOMACRO
	AHNHS_ACTAPC_DETECT_SPEEDHACK
	AHNHS_ENGINE_DETECT_GAME_HACK
	AHNHS_ACTAPC_DETECT_ABNORMAL_MEMORY_ACCESS
	AHNHS_ACTAPC_DETECT_ENGINERUNNING
	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_ACTAPC_DETECT_RMEM MODIFY_FROM_LMP

AHNHS_CHKOPT_ANITFR EESEVER	AHNHS_ACTAPC_DETECT_ANTIFREESERVER
AHNHS_CHKOPT_ABNOR MAL_FUNCTION_CALL_V2	AHNHS_ACTAPC_DETECT_ABNORMAL_FUNCTION_CALL

### ⚠ 주의

만약 위 콜백 코드 가운데 처리되지 않은 콜백 코드가 단 한 건이라도 있는 경우, HackShield가 실행 중인 게임을 비정상적으로 종료시킬 수 있습니다.

AHNHS\_CHKOPT\_SELF\_DESTRUCTION 옵션 예제는 다음과 같습니다.

```
int __stdcall AhnHS_Callback(long lCode, long lParamSize, void* pParam)
{
    switch(lCode)
    {
        // Engine 콜백
        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_ENGINERUNNING:
        case AHNHS_ACTAPC_DETECT_AUTOMACRO:
        case AHNHS_ACTAPC_DETECT_CODEMISMATCH:
        case AHNHS_ACTAPC_DETECT_MEM MODIFY_FROM_LMP:
        case AHNHS_ACTAPC_DETECT_RMEM MODIFY_FROM_LMP:
        case AHNHS_ACTAPC_DETECT_ANTIFREESERVER:
        case AHNHS_ACTAPC_DETECT_ABNORMAL_HACKSHIELD_STATUS:
        case AHNHS_ACTAPC_DETECT_ABNORMAL_FUNCTION_CALL:
            MessageBox ( );
            ExitGame ( );
            break;

    }

    return 1;
}
```

그림 9 AHNHS\_CHKOPT\_SELF\_DESTRUCTION 옵션 예제

### AHNHS\_DISPLAY\_HACKSHIELD\_TRAYICON

HackShield 실행 시 작업 표시줄에 HackShield 트레이 아이콘을 보여주는 옵션입니다.

### AHNHS\_CHKOPT\_DETECT\_VIRTUAL\_MACHINE

HackShield가 가상 머신이나 에뮬레이터에서 실행되는 것을 방지하는 옵션입니다. 이 옵션을 적용하면 이미 정의되어 있는 가상 머신이나 에뮬레이터에서 게임이 실행 되는 것을 방지할 수 있습니다. 이 기능은 가상 머신의 기능 변경에 의하여 감지 로직이 추가, 또는 변경될 수 있습니다.

---

 참고

AHNHS\_CHKOPT\_DETECT\_VIRTUAL\_MACHINE 옵션으로 감지 가능한 가상 머신 프로그램은 Virtual PC, VMWare Workstation, Virtual Box, Parallels Workstation입니다. 단, Virtual Box, Parallels Workstation의 경우 9X 계열의 Guest 머신에 대한 감지는 하지 않습니다.

---

#### AHNHS\_CHKOPT\_UPDATED\_FILE\_CHECK

HackShield 업데이트가 완료되면 업데이트된 파일들을 이용하여 HackShield가 정상적으로 동작하고 있는지 확인하는 옵션입니다.

이 옵션을 사용하기 위한 제약 사항은 다음과 같습니다.

- HackShield 기동 경로 내에 HackShield 업데이트 관련 모듈 V3Hunt.dll 이 반드시 존재
- HackShield 기동 경로 내에 HackShield 업데이트 관련 파일 HSUpdate.env 가 반드시 존재
- HSUpdate.env 파일 내에 기록되는 업데이트 서버 주소는 HTTP 만 지원

---

 주의

HSUpdate.env 파일 내에 FTP를 지원하는 주소가 기록되어 있으면 해당 옵션으로 기능을 실행하는 중에 에러가 발생하여 게임 클라이언트로 콜백이 전송될 수 있습니다.

---

---

 참고

AHNHS\_CHKOPT\_UPDATED\_FILE\_CHECK 옵션 사용 시 다음 기능의 사용을 권장합니다.  
- AhnHS\_HSUpdateEx()를 통한 업데이트 기능을 실행. 해당 함수 호출 시 '게임 코드'를 인자로 전달해야 함  
- HSUpSetEnv.exe 툴을 통한 HSUpdate.env 파일에 '게임 코드' 저장 후 배포

---

#### AHNHS\_CHKOPT\_ABNORMAL\_FUNCTION\_CALL\_V2

HackShield가 적용된 게임의 실행 파일이나 모듈에서 보호하려는 함수가 해킹툴에 의해 호출되는 것을 감지하는 옵션입니다.

---

 참고

이 옵션을 사용하는 방법은 [2.3.10 \\_AhnHS\\_VerifyProtectedFunction\(\)](#)을 참고하십시오.

---

#### AHNHS\_CHKOPT\_SEND\_MONITOR\_ONCE

모니터링 전송과 관련된 역할을 하는 옵션입니다. 이 옵션을 사용하면 중복된 로그는 모니터링 서버로 한 번만 전송됩니다. 예를 들어 Hack.exe 해킹툴을 HackShield가 여러 번 감지해도 로그는 모니터링 서버로 한 번만 전송됩니다. 또한 이 옵션은 게임 유저 ID가 입력되지 않으면 게임 종료 전까지 모니터링 서버로 로그를 전송하지 않는 역할도 합니다. 단, 게임 종료 순간까지도 게임 유저 ID가 입력되지 않으면, 게임 유저 ID 없이 모니터링

서버로 로그를 전송합니다.

---

 참고

게임 유저 ID 입력 상태는 `_AhnHS_SetUserId()` 호출 여부로 알 수 있습니다.

---

**표 6 HackShield 옵션별 콜백 코드 및 return value**

적용 옵션	콜백 코드 및 return Value
AHNHS_CHKOPT_SPEEDHACK	AHNHS_ACTAPC_DETECT_SPEEDHACK
AHNHS_CHKOPT_KDTRACER	AHNHS_ACTAPC_DETECT_KDTRACE
AHNHS_CHKOPT_AUTOMOUSE	AHNHS_ACTAPC_DETECT_AUTOMACRO
AHNHS_CHKOPT_AUTOMOUSE	AHNHS_ACTAPC_DETECT_AUTOMACRO
AHNHS_CHKOPT_LOCAL_MEMO RY_PROTECTION	AHNHS_ACTAPC_DETECT_MEM MODIFY_FROM_LMP
AHNHS_CHKOPT_LOCAL_MEMO RY_PROTECTION	AHNHS_ACTAPC_DETECT_RMEM MODIFY_FROM_LMP
AHNHS_CHKOPT_ANTIFREESERV ER	AHNHS_ACTAPC_DETECT_ANTIFREESERVER
AHNHS_CHKOPT_ABNORMAL_FU NCTION_CALL	AHNHS_ACTAPC_DETECT_ABNORMAL_FUNCTION_CALL
AHNHS_CHKOPT_ABNORMAL_FU NCTION_CALL_V2	
AHNHS_CHKOPT_READWRITEPR OCESSEMMORY	AHNHS_ACTAPC_DETECT_ABNORMAL_MEMORY_ACCESS
AHNHS_CHKOPT_PROCESSSCAN	AHNHS_ENGINE_DETECT_GAME_HACK AHNHS_ENGINE_DETECT_WINDOWED_HACK
AHNHS_CHKOPT_UPDATED_FILE _CHECK	AHNHS_ACTAPC_DETECT_ABNORMAL_HACKSHIELD_STATUS
AHNHS_CHKOPT_STANDALONE	AHNHS_ACTAPC_DETECT_ABNORMAL_HACKSHIELD_STATUS Error code return (HS_ERR_ALREADY_GAME_STARTED)
AHNHS_CHKOPT_PROTECT_D3DX	AHNHS_ACTAPC_DETECT_HOOKFUNCTION
AHNHS_CHKOPT_SELF_DESTRU CTION	AHNHS_ACTAPC_DETECT_SELF_DESTRUCTION
AHNHS_CHKOPT_DETECT_VIRTU AL_MACHINE	Error code return (HS_ERR_VIRTUAL_MACHINE_DETECT)

---

 참고

위 표는 대부분 HackShield의 감지 기능에 해당하는 목록이며, 그 외 방어 기능은 내부적으로 처리함으로써 콜백 코드 및 return value를 게임에 전달하지 않습니다.

---

## SPEEDHACK SENSING RATIO

### AHNHS\_SPEEDHACK\_SENSING\_RATIO\_HIGHEST

게임 시간을 빠르게 변경하여 게임 진행에 영향을 주는 해킹툴을 감지할 때 사용하는 옵

션입니다. 가장 민감하게 동작하는 레벨로서 기준치는 36.5%와 -3.0%입니다. A Speeder 스피드 핵 프로그램을 기준으로 봤을 때 0.7320배 이하로 속도를 느리게 하거나 1.0718배 이상으로 속도를 빠르게 할 때 스피드 핵으로 진단합니다.

#### **AHNHS\_SPEEDHACK\_SENSING\_RATIO\_HIGH**

게임 시간을 빠르게 변경하여 게임 진행에 영향을 주는 해킹툴을 감지할 때 사용하는 옵션입니다. 두 번째로 민감하게 동작하는 레벨로서 기준치는 30.5%와 -9.5%입니다. A Speeder 스피드 핵 프로그램을 기준으로 봤을 때 0.7579배 이하로 속도를 느리게 하거나 1.1487배 이상으로 속도를 빠르게 할 때 스피드 핵으로 진단합니다.

#### **AHNHS\_SPEEDHACK\_SENSING\_RATIO\_NORMAL**

게임 시간을 변경하여 게임 진행에 영향을 주는 해킹툴을 감지할 때 사용하는 옵션입니다. 일반적인 경우에 동작하는 레벨로서 기준치는 26.0%와 -12.5%입니다. A Speeder 스피드 핵 프로그램을 기준으로 봤을 때 0.7846배 이하로 속도를 느리게 하거나 1.1892배 이상으로 속도를 빠르게 할 때 스피드 핵으로 진단합니다.

#### **AHNHS\_SPEEDHACK\_SENSING\_RATIO\_LOW**

게임 시간을 느리게 변경하여 게임 진행에 영향을 주는 해킹툴을 감지할 때 사용하는 옵션입니다. 두 번째로 둔감하게 동작하는 레벨로서 기준치는 22.5%와 -15.5%입니다. A Speeder 스피드 핵 프로그램을 기준으로 봤을 때 0.8123배 이하로 속도를 느리게 하거나 1.2311배 이상으로 속도를 빠르게 할 때 스피드 핵으로 진단합니다.

#### **AHNHS\_SPEEDHACK\_SENSING\_RATIO\_LOWEST**

게임 시간을 느리게 변경하여 게임 진행에 영향을 주는 해킹툴을 감지할 때 사용하는 옵션입니다. 가장 둔감하게 동작하는 레벨로서 기준치는 17.5%와 -18.5%입니다. A Speeder 스피드 핵 프로그램을 기준으로 봤을 때 0.7320배 이하로 속도를 느리게 하거나 1.2746배 이상으로 속도를 빠르게 할 때 스피드 핵으로 진단합니다.

#### **AHNHS\_SPEEDHACK\_EXOPT\_ALLOW\_SLOW**

게임의 시간을 느리게 변경하는 해킹툴에 대해서 감지를 하지 않도록 하는 옵션입니다. 게임의 시간을 느리게 변경하는 행위가 게임진행에 영향을 주지 않을 경우 사용할 수 있습니다. 단, SPEEDHACK\_SENSING\_RATIO 옵션과 함께 사용해야 하며 단독으로 사용할 수 없습니다. AHNHS\_SPEEDHACK\_EXOPT\_ALLOW\_SLOW 사용 예제는 다음과 같습니다.

```
UINT unAdditionalRatio = AHNHS_SPEEDHACK_SENSING_RATIO_HIGHEST |  
                        AHNHS_SPEEDHACK_EXOPT_ALLOW_SLOW
```

**그림 10 AHNHS\_SPEEDHACK\_EXOPT\_ALLOW\_SLOW 사용 예제**

## **SELF-DESTRUCTION TIME RATIO**

### **AHNHS\_SELFDESTRUCTION\_RATIO\_FAST**

해킹툴을 감지하여 빠르게 게임을 종료해야 하는 상황에 사용하는 옵션입니다. 가장 빠르게 종료 되는 옵션으로 해킹툴 감지 30초 후에 게임이 종료 됩니다.

### **AHNHS\_SELFDESTRUCTION\_RATIO\_NORMAL**

해킹툴을 감지한 후에 게임을 종료 시키는 옵션입니다. 1분 후에 게임이 종료되는 옵션으로 특별한 경우가 아니라면 이 옵션을 사용합니다. 해당 옵션을 설정 하지 않았을 경우 기본적으로 이 레벨이 적용됩니다.

### **AHNHS\_SELFDESTRUCTION\_RATIO\_SLOW**

해킹툴을 감지한 후에 게임 종료 전에 게임사에서 특정작업을 수행하려고 할 경우 적용 시키는 옵션입니다. 해당옵션이 적용되면 게임은 3분 후에 종료됩니다.

### **AHNHS\_SELFDESTRUCTION\_RATIO\_VERYSLOW**

해킹툴을 감지한 후에 게임 종료 전에 게임사에서 특정작업을 수행하려고 할 경우 적용 시키는 옵션으로 가장 느리게 게임을 종료시키는 옵션입니다. 해당 옵션이 적용되면 게임은 5분 후에 종료됩니다.

```
UINT unAdditionalRatio = AHNHS_SPEEDHACK_SENSING_RATIO_HIGHEST |  
                         AHNHS_SELFDESTRUCTION_RATIO_NORMAL);  
  
nRet = _AhnHS_Initialize ( //  
                           szFullPath,  
                           HS_CallbackProc,           // 콜백 함수  
                           1000,                      // 게임 코드  
                           "B228F291B7D7FAD361D7A4B7", // 라이선스 키  
                           dwOption,                  // 옵션 플래그  
                           unAdditionalRatio  
                         );
```

**그림 11 SELF-DESTRUCTION TIME RATIO 사용 예제**

---

#### **참고**

해당 핵쉴드 자체종료 대기 시간 레벨(SELF-DESTRUCTION TIME RATIO)은 AHNHS\_CHKOPT\_SEND\_MONITOR\_DELAY 옵션이 설정되어 있고 게임 유저 ID가 입력되지 않았을 경우 게임 유저 ID가 입력될 때까지 일정 시간 대기한 다음 핵쉴드 자체 종료 대기 시간이 적용이 됩니다.

---

## **Return Value**

### **HS\_ERR\_OK (Value = 0x0000)**

- 설명: 함수 호출 성공

- 원인: 정상적인 경우
- 확인사항: N/A

#### **HS\_ERR\_INVALID\_PARAM(Value = 0x002)**

- 설명: 잘못된 파라미터
- 원인: 콜백 함수의 포인터가 잘못 설정됐거나 라이선스 키 값이 NULL 인 경우
- 확인사항
  - szFileName 값의 파일 경로가 정확한지 확인
  - unSHackSensingRatio 값에 Sensing Ratio 옵션을 적용했는지 확인  
(값이 0 이거나 Extra 옵션 값만 적용한 경우 발생할 수 있음)

#### **HS\_ERR\_NOT\_INITIALIZED(Value = 0x003)**

- 설명: 초기화 실패
- 원인: HackShield 동작 시 사용하는 시스템 라이브러리가 정상적으로 로드되지 않은 경우
- 확인사항: 지속적인 에러 발생 시 안랩 HackShield 기술 지원에 문의

#### **HS\_ERR\_COMPATIBILITY\_MODE\_RUNNING (Value = 0x004)**

- 설명: 호환성 모드로 게임 실행
- 원인: 게임 클라이언트가 Windows XP 계열에서 제공하는 호환성 모드인 Compatibility Mode로 실행한 경우
- 확인사항: 호환성 모드로 게임을 실행했다면 악의적인 목적을 가진 사용자라고 판단되므로 프로그램을 강제 종료하고 다시 시작해야 함

#### **HS\_ERR\_EXCEPTION\_RAISED (Value = 0x007)**

- 설명: 예외(Exception) 발생
- 원인: HackShield 동작 중 예외(Exception)가 발생한 경우
- 확인사항: HShield.log 와 수집한 AhnReport 를 안랩 HackShield 기술 지원에 전달

#### **HS\_ERR\_INVALID\_LICENSE (Value = 0x100)**

- 설명: 잘못된 라이선스 키 입력
- 원인: 초기화 함수 파라미터인 게임 코드와 라이선스 키 값이 실제 값과 일치하지 않는 경우
- 확인사항: 라이선스 키값이 정확히 입력됐는지 확인

#### **HS\_ERR\_INVALID\_FILES (Value = 0x101)**

- 설명: 잘못된 HackShield 파일
- 원인: 인터페이스 DLL 파일인 EhSvc.dll 이 위/변조됐거나, 버전이 같지 않을 경우. 인터페이스 DLL 파일이 위/변조되면 해킹 차단 기능이 무력화될 수 있음
- 확인사항: HackShield 파일이 구 버전이거나 HackShield 폴더에 파일이 있는지 확인

### **HS\_ERR\_INIT\_DRV\_FAILED (Value = 0x102)**

- 설명: HackShield 드라이버 시작 실패
- 원인: 해킹 차단 기능 드라이버가 초기화에 실패한 경우. 드라이버 초기화에 실패하면 해킹 차단 기능이 정상 동작할 수 없으므로 프로그램을 강제 종료하고 다시 시작해야 함
- 확인사항
  - 시스템 폴더에 EagleXNT.sys 나 EagleX64.sys 파일의 권한이 변경돼 접근할 수 없는 형태가 됐는지 확인
  - EagleXNT.sys 나 EagleX64.sys 파일이 언로드되지 않고 떠 있는지 확인

### **HS\_ERR\_ALREADY\_INITIALIZED (Value = 0x104)**

- 설명: HackShield 이미 초기화
- 원인: \_AhnHS\_Initialize()를 호출하여 시스템이 이미 초기화되어 있는 상태인 경우
- 확인사항: \_AhnHS\_Initialize()는 프로그램 초기화 시점에 한 번만 호출할 수 있으며 중복으로 호출할 수 없기 때문에 내부적으로 위 함수를 여러 번 호출하지 않았는지 확인

### **HS\_ERR\_DEBUGGER\_DETECT (Value = 0x105)**

- 설명: 디버거 감지
- 원인: 시스템에서 디버거가 실행 중인 경우
- 확인사항
  - 개발 과정에 있다면 릴리즈 버전이 아닌 Dev 버전을 사용하여 디버깅(Remarks 참고)
  - 사용자에게 해당 에러가 발생한다면 해킹 시도가 의심되므로 게임을 종료하고 디버거를 중지한 다음 다시 실행. 만약, 디버거가 없는데도 해당 에러 발생 시 안랩 HackShield 기술 지원에 문의

### **HS\_ERR\_NEED\_ADMIN\_RIGHTS (Value = 0x107)**

- 설명: 관리자 계정 필요
- 원인: 게임 클라이언트가 Windows NT 계열의 시스템에서 관리자 계정이 아닌 일반 사용자 계정의 권한으로 실행된 경우
- 확인사항: 유저 계정에서 실행했다면, 쉐도우 계정이 만들어 졌는지 확인. 일반 사용자 계정 권한으로 HackShield 가 제공하는 기능을 사용하려면 HackShield 쉐도우 계정을 먼저 생성



#### **참고**

자세한 내용은 [8장](#)을 참고하십시오.

---

### **HS\_ERR\_MODULE\_INIT\_FAILED (Value = 0x108)**

- 설명: HackShield 모듈 초기화 실패
- 원인: HackShield 구동에 필요한 초기화 작업에서 문제가 발생한 경우

- 확인사항: HShield.log 와 수집한 AhnReport 를 안랩 HackShield 기술 지원에 전달

#### **HS\_ERR\_UNKNOWN (Value = 0x001)**

- 설명: 알 수 없는 에러
- 원인: 함수 내부에서 예외가 발생했거나, 함수 구조적인 문제일 가능성이 큰 경우
- 확인사항: HShield.log 와 수집한 AhnReport 를 안랩 HackShield 기술 지원에 전달

#### **Remarks**

개발이나 테스트 과정에서 게임 클라이언트를 디버깅하려면 개발자용 HShield.lib, Ehvsc.dll, hshield.dat를 사용해야 합니다.

---

##### **⚠ 주의**

Ehsvc.dll과 hshield.dat를 릴리즈 버전과 함께 사용하면 서버 연동 시 문제가 발생할 수 있습니다.

---

경로는 `\Developer\`이며, 개발자용 모듈을 사용하면 HackShield 실행 시 다음과 같은 로고가 나타나고, 해당 로고를 누를 경우 사라집니다.

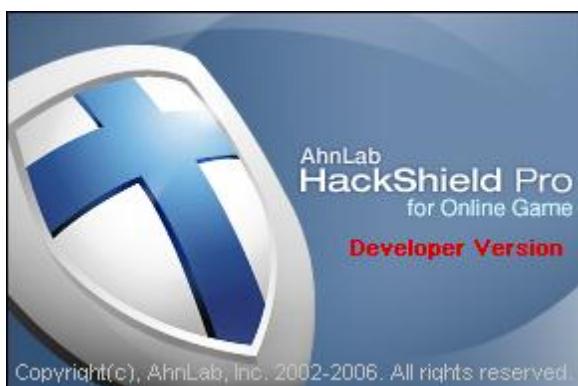


그림 12 HackShield 로고 화면

#### **2.3.2 \_AhnHS\_Callback()**

해킹 차단 및 해킹툴 탐지 결과에 대한 정보를 전달하는 콜백 함수입니다. 이 함수가 전달하는 이벤트는 다음과 같습니다.

- 해킹 차단 기능과 관련된 이벤트
- 해킹툴 탐지 기능과 관련된 이벤트

```
int __stdcall AhnHS_Callback(
    long lCode,
    long lParamSize,
    void* pParam
);
```

## Parameters

Parameter	Value	Description
lCode	Long	이벤트 코드
lParamSize	Long	이벤트 파라미터의 크기
pParam	Void*	이벤트 파라미터

## Events

### AHNHS\_ENGINE\_DETECT\_GAME\_HACK

시스템에서 실행하고 있는 게임 전용 해킹툴을 탐지한 경우 발생하는 이벤트입니다. 이벤트를 통해 전달된 해킹툴 정보를 사용하여 고객은 해킹툴의 실행 파일을 강제로 삭제하는 등의 조치를 취할 수 있습니다.

- pParam: 탐지된 게임 전용 해킹툴의 실행 파일 이름과 파일 경로
- lParamSize: 탐지된 해킹툴 실행 파일 이름의 문자 길이

### AHNHS\_ENGINE\_DETECT\_WINDOWED\_HACK

전체화면으로 실행되는 게임에 대하여 강제로 창 모드로 실행시키는 해킹툴을 탐지한 경우 발생하는 이벤트입니다. 이 이벤트는 창 모드로 실행이 불가능한 게임에 대해서만 적용되는 사항이기 때문에 전체 화면 모드와 창 모드를 모두 지원하는 게임에서는 사용하지 마십시오.

- pParam: NULL
- lParamSize: 0

### AHNHS\_ACTAPC\_DETECT\_ALREADYHOOKED

해킹 차단 기능이 실행 중일 때, 해킹 차단 기능에서 보호하고자 하는 API 함수가 이미 후킹되어 있을 경우 발생하는 이벤트입니다. 해킹 툴이 아닌 정상 프로그램에서도 기능 구현을 위해서 후킹을 할 수 있으므로 해킹 툴 인지에 대한 판단은 게임 개발사 정책대로 결정해야 합니다.

- pParam: ACTAPCPARAM\_DETECT\_HOOKFUNCTION\*
- lParamSize: ACTAPCPARAM\_DETECT\_HOOKFUNCTION 구조체의 길이

### AHNHS\_ACTAPC\_DETECT\_HOOKFUNCTION

해킹 차단 기능이 동작 중일 때 Win32 함수나 보호 함수가 후킹됐을 경우 발생하는 이벤트입니다. 이 이벤트가 발생한 경우 게임 클라이언트에 해킹툴이 침입했을 확률이 높기 때문에 게임을 강제 종료해야 합니다.

- pParam: ACTAPCPARAM\_DETECT\_HOOKFUNCTION\*
- lParamSize: ACTAPCPARAM\_DETECT\_HOOKFUNCTION 구조체의 길이

```
struct _ACTAPCPARAM_DETECT_HOOKFUNCTION
{
```

```

        char szFunctionName[128];
        char szModuleName[128];
    } ACTAPCPARAM_DETECT_HOOKFUNCTION, *PACTAPCPARAM_DETECT_HOOKFUNCTION;

```

- szFunctionName: 후킹된 함수 이름
- szModuleName: 후킹한 모듈 이름

### AHNHS\_ACTAPC\_DETECT\_AUTOMOUSE

오토 마우스 종류의 프로그램으로 키보드나 마우스를 조정하여 데이터를 자동 입력할 경우 발생하는 이벤트입니다.

- pParam: ACTAPCPARAM\_DETECT\_AUTOMOUSE
- IParamSize: ACTAPCPARAM\_DETECT\_AUTOMOUSE 크기

```

typedef struct
{
    BYTE   byDetectType;
    DWORD  dwPID;
    CHAR   szProcessName[16+1];
    CHAR   szAPIName[128];
} ACTAPCPARAM_DETECT_AUTOMOUSE, *PACTAPCPARAM_DETECT_AUTOMOUSE;

```

위 예제에서, dwPID, szProcessName, szAPIName은 현재 사용하지 않으며 byDetectType의 값과 의미는 다음과 같습니다.

- EAGLE\_AUTOMOUSE\_APCTYPE\_SHAREDMEMORY\_ALTERATION (3): 오토마우스 차단 기능을 하는 라이브러리의 내부 데이터 변경. HackShield가 해킹 차단 시 사용하는 내부 정보를 해킹 프로그램이 수정하면 차단 기능이 동작되지 않으므로 게임 프로그램을 강제 종료시켜야 함
- EAGLE\_AUTOMOUSE\_APCTYPE\_API\_CALLED: 키보드와 마우스 관련 API 호출
- EAGLE\_AUTOMOUSE\_APCTYPE\_API\_ALTERATION: API 후킹 변조

### AHNHS\_ACTAPC\_DETECT\_AUTOMACRO

오토 마우스 종류의 프로그램을 사용해서 키보드나 마우스를 조정하여 자동으로 데이터를 입력할 경우 발생하는 이벤트입니다. 비정상적인 입력 값이 반복해서 나타나는 현상으로 게임을 강제 종료시켜야 합니다.

- pParam: ACTAPCPARAM\_DETECT\_AUTOMACRO
- IParamSize: ACTAPCPARAM\_DETECT\_AUTOMACRO 의 크기

```

typedef struct
{
    BYTE   byDetectType;
    CHAR   szModuleName[128];

} ACTAPCPARAM_DETECT_AUTOMACRO,
*PACTAPCPARAM_DETECT_AUTOMACRO;

```

```
#define EAGLE_AUTOMACRO_APCTYPE_KEYBOARD      1  
#define EAGLE_AUTOMACRO_APCTYPE_MOUSE        2
```

위 예제에서, szModuleName는 현재 사용하지 않으며 byDetectType의 값과 의미는 다음과 같습니다.

- EAGLE\_AUTOMACRO\_APCTYPE\_KEYBOARD: 마우스 관련 오토 마우스
- EAGLE\_AUTOMACRO\_APCTYPE\_MOUSE: 키보드 관련 오토마우스

#### **AHNHS\_ACTAPC\_DETECT\_DRIVERFAILED**

해킹 차단 드라이버가 시스템에 로드되지 않을 경우 발생하는 이벤트입니다. 해킹 차단 드라이버가 시스템에 로드되지 않으면 해킹 차단 기능이 정상 동작하지 않으므로 즉시 게임을 강제 종료시켜야 합니다. 또한 비정상적으로 드라이버가 제거되면 시스템이 불안정해질 수 있으므로 시스템도 다시 시작해야 합니다.

- pParam: NULL
- IParamSize: 0

#### **AHNHS\_ACTAPC\_DETECT\_SPEEDHACK**

시스템의 시간 변화 속도가 비정상적일 경우 발생하는 이벤트입니다. 하드웨어나 소프트웨어 방식의 스피드 핵이 실행 중일 가능성이 매우 높으므로 시간에 민감한 게임은 강제 종료시켜야 합니다. 하드웨어 방식의 스피드 핵은 시스템의 시간을 제어하는 하드웨어를 직접 조작하여 시스템 전체 시간에 영향을 미칩니다. 이러한 방식의 하드웨어 스피드 핵은 Windows 버전에 따라 운영체제에서 차단하기도 합니다.

- pParam: (double \*) 최근 5 초 동안의 시간 데이터 정보
- IParamSize: pParam 으로 전달된 시간 데이터의 개수

#### **AHNHS\_ACTAPC\_DETECT\_KDTRACE**

커널 레벨이나 애플리케이션 레벨의 디버거에 의해 디버거 트레이스가 발생하면 전달되는 이벤트입니다. 이 이벤트가 발생하면 해커와 같은 악의적인 사용자가 게임을 디버깅하고 있을 가능성이 높으므로 게임을 종료하는 것이 좋습니다.

- pParam: NULL
- IParamSize: 0

#### **AHNHS\_ACTAPC\_DETECT\_ABNORMAL\_MEMORY\_ACCESS**

허가되지 않은 프로세스가 게임 프로세스의 메모리에 접근할 때 발생하는 이벤트입니다. 메모리 접근을 시도한 프로세스의 경로와 이름을 알고 있으므로 어떤 프로세스가 접근했는지 확인할 수 있습니다.

- pParam: 탐지된 게임 전용 해킹툴의 실행 파일 이름과 파일 경로
- IParamSize: 탐지된 해킹툴 실행 파일 이름의 문자 길이

#### **AHNHS\_ACTAPC\_DETECT\_KDTRACE\_CHANGED**

디버깅 시도 차단 루틴이 변경된 경우 발생하는 이벤트입니다. 이 이벤트가 발생하면 게임 프로그램에 대한 디버깅 시도 차단 루틴이 수행되는 동안 커널 레벨의 디버거 프로그램이 활성화된 것으로 게임 프로그램을 강제 종료해야 합니다.

- pParam: NULL
- IParamSize: 0

#### **AHNHS\_ACTAPC\_DETECT\_MODULE\_CHANGE**

HackShield 모듈에 대한 위/변조를 감지한 경우 발생하는 이벤트입니다. 이 이벤트가 발생하면 HackShield가 정상적으로 동작하지 않을 수 있으므로 게임 프로그램을 강제 종료해야 합니다.

- pParam: NULL
- IParamSize: 0

#### **AHNHS\_ACTAPC\_DETECT\_ENGINERUNNING**

HackShield의 휴리스틱 엔진 파일인 3n.mhe가 삭제됐거나 비정상적이어서 로드가 안됐을 경우 발생하는 이벤트입니다. 이 이벤트가 발생하면 HackShield의 ProcessScan 기능이 정상적으로 동작하지 않을 수 있으므로 게임을 강제 종료해야 합니다.

- pParam: NULL
- IParamSize: 0

#### **AHNHS\_ACTAPC\_DETECT\_CODEMISMATCH**

HackShield의 인터페이스 모듈인 Ehsvc.dll의 코드 영역이 조작됐을 경우 발생하는 이벤트입니다. 이 이벤트가 발생하면 HackShield의 기능이 정상적으로 동작하지 않을 수 있으므로 게임을 강제 종료해야 합니다.

- pParam: NULL
- IParamSize: 0

#### **AHNHS\_ACTAPC\_DETECT\_MEM\_MODIFY\_FROM\_LMP**

보호 대상으로 지정한 파일의 메모리 영역이 조작됐을 경우 발생하는 이벤트입니다. 이 이벤트가 발생하면 해당 모듈이 정상적으로 동작하지 않을 수 있으므로 게임을 강제 종료해야 합니다.

- pParam: 조작된 모듈 이름이거나 조작된 페이지 시작 주소
- IParamSize: 0

#### **AHNHS\_ACTAPC\_DETECT\_LMP\_FAILED**

LMP 보호 기능이 해킹 및 시스템 이상과 같은 외부 요인으로 인해 정상 동작하지 않을 경우 발생하는 이벤트입니다. 게임을 다시 실행하거나 시스템을 다시 시작하십시오.

- IParamSize: 0

### AHNHS\_ACTAPC\_DETECT\_RMEM MODIFY\_FROM\_LMP

보호 대상으로 지정한 파일의 읽기전용 메모리 영역이 조작됐을 경우 발생하는 이벤트입니다. 이 이벤트가 발생하면 해당 모듈이 정상적으로 동작하지 않을 수 있으므로 게임을 강제 종료해야 합니다.

- pParam: 조작된 페이지 시작 주소
- IParamSize: 0

### AHNHS\_ACTAPC\_DETECT\_ANTIFREESERVER

게임에서 비정상적인 주소의 서버로 접속하려 할 때 발생하는 이벤트입니다. 이 경우 정상적인 게임 실행이 아니므로 게임을 강제 종료해야 합니다. pParam은 접속하려는 주소 값입니다. 이 주소 값은 내부 확인용으로만 사용하고 사용자에게 보여주지 않는 것을 권장합니다.

- pParam: 접속하려는 IP 주소
- IParamSize: 0

### AHNHS\_ACTAPC\_DETECT\_ABNORMAL\_HACKSHIELD\_STATUS

HackShield 동작이 정상적으로 이루어지지 않았을 경우 발생하는 이벤트입니다. 이 이벤트가 발생하면 HackShield가 정상적으로 동작하지 않고 있으므로 게임을 강제 종료해야 합니다.

- pParam: NULL
- IParamSize: 0

### AHNHS\_ACTAPC\_STATUS\_HACKSHIELD\_RUNNING

HackShield가 정상적으로 동작하고 있음을 나타내는 이벤트입니다. \_AhnHS\_CheckHackShieldRunningStatus()를 호출하여 해당 기능을 시작합니다. HackShield가 정상적으로 동작하고 있는 경우 HackShield의 \_AhnHS\_StopService(), \_AhnHS\_UnInitialize()가 호출 되기 전까지 약 25초를 주기로 이 이벤트가 발생합니다. 이 이벤트가 주기적으로 발생되지 않는다면 HackShield가 정상 동작하고 있지 않는 것으로 판단합니다.

- pParam: (DWORD \*) HackShield 동작 상태 값(enum HS\_RUNNING\_STATUS에 정의된 값 중 하나)
- IParamSize: pParam의 크기

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

---

#### ⚠ 주의

AHNHS\_ACTAPC\_STATUS\_HACKSHIELD\_RUNNING은 HackShield 내부적으로 25초를 주기로 호출됩니다. 시스템 상황에 따라 25초 내외를 콜백 주기로 사용하십시오.

### 2.3.3 \_AhnHS\_StartService()

해킹툴 탐지 기능과 해킹 차단 기능을 동작하게 하는 함수입니다.  
\_AhnHS\_Initialize()를 호출한 다음에 호출해야 하며 중복으로 호출할 수 없습니다.  
\_AhnHS\_StopService()를 호출하여 HackShield를 중지한 경우, 이 함수를 다시 호출하여 HackShield를 다시 시작할 수 있습니다.

```
EHsvc_API  
int __stdcall  
_AhnHS_StartService();
```

#### Parameters

N/A

#### Return Value

##### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

##### HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize()를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

##### HS\_ERR\_START\_ENGINE\_FAILED (Value = 0x200)

- 설명: 엔진 로드 실패
- 원인: 초기화 함수를 호출할 때 해킹툴 프로세스 감지 옵션(AHNHS\_CHKOPT\_PROCESSSCAN)을 설정했으나 해킹툴 패턴 엔진 초기화에 실패한 경우, 해킹툴 패턴 엔진 관련 파일이 제대로 설치되지 않았거나, 패턴 엔진 관련 파일이 HackShield 폴더에 제대로 설치되지 않았을 경우
- 확인사항: 프로그램을 강제 종료하고 다시 시작하거나 다시 설치

##### HS\_ERR\_ALREADY\_SERVICE\_RUNNING (Value = 0x201)

- 설명: HackShield 가 이미 실행 중

- 원인: \_AhnHS\_StartService()를 이미 호출한 상태에서 다시 호출한 경우
- 확인사항: 이 함수를 다시 호출하려면 반드시 \_AhnHS\_StopService()를 호출해서 HackShield 를 종료해야 하는데 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

#### **HS\_ERR\_DRV\_FILE\_CREATE\_FAILED (Value = 0x202)**

- 설명: HackShield 드라이브 파일 생성 실패
- 원인: 해킹 차단을 위한 드라이버 파일 생성에 실패한 경우. HackShield 프로그램 내부에서는 해킹 차단 처리를 위한 드라이버를 게임이 시작하는 시점에 생성하여 로드시키는데 게임 시작 시 드라이버 파일 생성에 문제가 발생한 경우 나타나는 에러
- 확인사항
  - 현재 세션이 시스템 폴더에 대하여 쓰기 권한이 있는지 확인
  - 시스템 폴더에 있는 HackShield 드라이버 파일 EagleXNT.sys 나 EagleX64.sys 의 속성이 읽기 전용이거나 액세스 가능하지 않은 속성인지 확인. 만약 이와 같은 상황이라면 해당 파일을 삭제하고 HackShield 를 다시 시작해야 하며 지속적인 에러 발생 시 안랩 HackShield 기술 지원에 문의

#### **HS\_ERR\_REG\_DRV\_FILE\_FAILED (Value = 0x203)**

- 설명: HackShield 드라이브 파일 등록 실패
- 원인: 해킹 차단을 위한 드라이버 파일을 시스템에 등록하는데 실패한 경우
- 확인사항: HackShield 가 정상적으로 동작할 수 없으므로 프로그램을 강제 종료하고 다시 시작하거나 다시 설치

#### **HS\_ERR\_START\_DRV\_FAILED (Value = 0x204)**

- 설명: HackShield 드라이브 구동 실패
- 원인: 게임 클라이언트가 여러 가지 원인으로 인하여 비정상적으로 종료되면서 \_AhnHS\_StopService()나 \_AhnHS\_Uninitialize()를 호출하지 못하여 기존에 로드했던 드라이버가 정지되거나 로드하지 않아야 하는 드라이버가 계속 로드되어 있는 상태에서 다시 게임을 실행하는 경우. 시스템 자체가 불안하거나 문제가 있을 수 있으므로 프로그램을 종료하고 시스템을 다시 시작
- 확인사항
  - 현재 세션이 시스템 폴더에 대하여 쓰기 권한이 있는지 확인
  - 시스템 폴더에 HackShield 드라이버 파일인 EagleXNT.sys 나 EagleX64.sys 속성이 읽기 전용이거나 액세스 가능하지 않은 속성으로 되어 있는지 확인. 만약 위와 같은 상황이라면 해당 파일을 삭제하고 HackShield 를 다시 시작해야 하며 지속적인 에러 발생 시 안랩 HackShield 기술 지원에 문의

#### **HS\_ERR\_ALREADY\_GAME\_STARTED (Value = 0x206)**

- 설명: 게임 이미 실행 중
- 원인: 초기화 함수를 호출할 때 HackShield 단독 실행  
옵션(AHNHS\_CHKOPT\_STANDALONE)을 적용한 경우, 동일한 게임 코드로 게임이 이미 실행 중인 경우
- 확인사항: 게임 프로세스는 하나만 실행할 수 있으므로 해당 프로그램을 종료

#### **HS\_ERR\_VIRTUAL\_MACHINE\_DETECT (Value = 0x207)**

- 설명: 게임이 가상 머신이거나 에뮬레이터 상에서 실행 중
- 원인: 초기화 함수를 호출할 때 가상 머신에서의 실행을 방지하는 옵션 AHNHS\_CHKOPT\_DETECT\_VIRTUAL\_MACHINE 을 적용한 경우
- 확인사항: HackShield에서 제어하는 가상 머신 상에서의 실행을 방지

### **2.3.4 \_AhnHS\_StopService()**

해킹 차단 기능과 해킹툴 탐지 기능을 정지시키는 함수입니다.

```
EHSCV_API
int __stdcall
_AhnHS_StopService();
```

#### **Parameters**

N/A

#### **Return Value**

##### **HS\_ERR\_OK (Value = 0x0000)**

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

##### **HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)**

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize()를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

##### **HS\_ERR\_SERVICE\_NOT\_RUNNING (Value = 0x301)**

- 설명: HackShield 가 시작되지 않음

- 원인: \_AhnHS\_StartService()를 호출하여 HackShield 를 시작하지 않은 상태에서 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

### 2.3.5 \_AhnHS\_Uninitialize()

시스템 내부적으로 사용되었던 메모리를 해제하고 변수를 초기화하는 함수입니다.

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

#### Parameters

N/A

#### Return Value

##### **HS\_ERR\_OK (Value = 0x0000)**

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

##### **HS\_ERR\_SERVICE\_STILL\_RUNNING (Value = 0x302)**

- 설명: HackShield 가 실행 중
- 원인: \_AhnHS\_StopService()를 호출하여 HackShield 를 종료하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

##### **HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)**

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize()를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

### 2.3.6 \_AhnHS\_PauseService()

HackShield가 제공하는 기능 중 일부분의 기능을 잠시 중단하는 함수입니다. 메시지 후킹 방어 기능 중 키보드 방어에 대해서만 처리합니다. 게임 실행 중에 결재 등을 하기 위해 Microsoft Internet Explorer의 웹페이지 방식으로 사용자 입력을 받는 경우에 키보드 방어 기능으로 사용자 입력이 동작되지 않는 것을 일시 중지시키는데 사용합니다. 일시 중지

후 반드시 `_AhnHS_ResumeService()`를 사용하여 다시 활성화해야 합니다.

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

### Parameters

Parameter	Value	Description
DwPauseOption	DWORD	현재는 AHNHS_CHKOPT_MESSAGEHOOK 옵션만 사용 가능 다른 옵션 전달 시 HS_ERR_INVALID_PARAM 리턴

### Return Value

#### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

#### HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)

- 설명: HackShield 가 초기화되지 않음
- 원인: `_AhnHS_Initialize()`를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

#### HS\_ERR\_SERVICE\_NOT\_RUNNING (Value = 0x301)

- 설명: HackShield 가 시작되지 않음
- 원인: `_AhnHS_StartService()`를 호출하여 HackShield 를 시작하지 않은 상태에서 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

#### HS\_ERR\_INVALID\_PARAM (Value = 0x002)

- 설명: 잘못된 파라미터
- 원인: dwPauseOption 값이 AHNHS\_CHKOPT\_MESSAGEHOOK 이 아닌 경우
- 확인사항: N/A

### 2.3.7 `_AhnHS_ResumeService()`

`_AhnHS_PauseService()`를 호출하여 잠시 중단했던 HackShield 기능을 다시 활성화하

는 함수입니다. 메시지 후킹 방어 기능 중 키보드 방어에 대해서만 처리합니다.

---

 참고

이 기능의 필요성에 대해서는 [2.3.6\\_AhnHS\\_PauseService\(\)](#)를 참고하십시오.

---

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

### Parameters

Parameter	Value	Description
dwResumeOption	DWORD	현재는 AHNHS_CHKOPT_MESSAGEHOOK 옵션만 사용 가능 다른 옵션 전달 시 HS_ERR_INVALID_PARAM 리턴

### Return Value

#### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

#### HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize()를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

#### HS\_ERR\_SERVICE\_NOT\_RUNNING (Value = 0x301)

- 설명: HackShield 가 시작되지 않음
- 원인: \_AhnHS\_StartService()를 호출하여 HackShield 를 시작하지 않은 상태에서 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

#### HS\_ERR\_INVALID\_PARAM (Value = 0x002)

- 설명: 잘못된 파라미터
- 원인: dwResumeOption 값이 AHNHS\_CHKOPT\_MESSAGEHOOK 이 아닌 경우
- 확인사항: N/A

### 2.3.8 \_AhnHS\_CheckHackShieldRunningStatus()

HackShield의 동작 상태를 주기적으로 확인하여 게임에 상태 콜백을 전달해 주는 기능을 활성화하는 함수입니다. 게임에서 HackShield의 정상 동작 여부를 확인해야할 경우, 이 함수를 호출합니다.

\_AhnHS\_Initialize(), \_AhnHS\_StartService()를 호출해서 HackShield가 시작된 뒤부터 \_AhnHS\_StopService(), \_AhnHS\_UnInitialize()를 호출해서 HackShield를 종료하기 전까지 동작합니다.

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

#### Parameters

N/A

#### Return Value

##### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우. HackShield 동작 상태를 주기적으로 확인하여 게임으로 콜백을 넘겨 줌
- 확인사항: N/A

##### HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)

- 설명: 초기화 실패
- 원인: \_AhnHS\_StartService()를 호출하여 HackShield를 시작하지 않은 상태에서 호출한 경우. HackShield 동작 시 사용하는 시스템 라이브러리가 정상적으로 로드되지 않아 발생함
- 확인사항: HackShield 가 시작되지 않았거나 해킹으로 인해 HackShield 가 정상 동작하지 않음

##### HS\_ERR\_INVALID\_FILES (Value = 0x101)

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize()를 호출하지 않았거나 함수 호출에 실패하여 HackShield를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: HackShield 가 시작되지 않았거나 해킹으로 인해 HackShield 가 정상 동작하지 않음

---

#### !

#### 참고

\_AhnHS\_CheckHackShieldRunningStatus()를 호출하면 해킹 감지 정보가 아닌 HackShield의 정상 동작 상태를 주기적으로 알려주는 AHNHS\_ACTAPC\_STATUS\_HACKSHIELD\_RUNNING 상태 콜백이

---

발생합니다. 고객은 보안 기능 강화의 목적으로 사용할 수 있습니다.

---

### 2.3.9 \_AhnHS\_SendHsLog()

HackShield 로그를 안랩 HackShield 기술 지원에 보내려고 하는 경우 HackShield 이벤트 코드, 게임유저 아이디, HackShield 경로와 함께 호출하는 함수입니다. 콜백 함수의 게임 종료 다이얼로그 박스에서 사용자가 선택적으로 로그를 보낼 수 있도록 로그 저장 버튼과 종료 버튼을 구현하여 로그 저장 버튼을 누르면 \_AhnHS\_SendHsLog()가 호출된 후 게임을 종료하도록 구현하십시오.

---

#### 참고

자세한 내용은 [2.2.3 프로그래밍 적용](#)을 참고하십시오.

---

```
void  
__stdcall  
_AhnHS_SendHsLog (  
    IN DWORD dwError,  
    IN const char* szUserID,  
    IN const char* szHShieldPath  
) ;
```

#### Parameters

Parameter	Value	Description
dwError	DWORD	HackShield 콜백 함수의 첫 번째 인자로 넘어오는 이벤트 코드(ICODE)를 입력
szUserID	const char*	게임 유저 아이디
szHShieldPath	const char*	Ehsvc.dll이 있는 HackShield 경로 ehsvc.dll이 제외된 폴더의 경로만 입력

#### Return Value

N/A

### 2.3.10 \_AhnHS\_VerifyProtectedFunction()

HackShield가 적용된 게임 모듈의 내부 함수가 외부의 모듈에 의해 호출되지 않도록 보호하는 함수입니다. 이 함수는 직접 사용하지 않고 반드시 헤더 파일에 정의된 매크로인 AHNHS\_PROTECT\_FUNCTION이거나 AHNHS\_PROTECT\_FUNCTIONEX를 사용해야 합니다. \_AhnHS\_StartService()가 정상적으로 호출된 이후에 기능이 동작합니다.

NT 이상에서만 동작하며 Windows 98은 지원하지 않습니다.

```
int  
__stdcall  
_AhnHS_VerifyProtectedFunction();
```

## Parameters

N/A

## Return Value

N/A

## Remarks

### ■ 옵션 처리

- \_AhnHS\_Initialize()에서 AHNHS\_CHKOPT\_ABNORMAL\_FUNCTION\_CALL\_V2 옵션 추가

```
nRet = _AhnHS_Initialize (      szFullPath  
                                HS_CallbackProc,           // 콜백 함수  
                                1000,                      // 게임 코드  
                                "B228F291B7D7FAD361D7A4B7" , // 라이선스 키  
AHNHS_CHKOPT_ALL|AHNHS_CHKOPT_ABNORMAL_FUNCTION_CALL_V2, // 옵션 플래그  
AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL);
```

- \_AhnHS\_Initialize(), \_AhnHS\_StartService()가 호출되지 않는 모듈에 대해서는 생략 가능

### ■ 보호함수에 매크로 추가

```
int CPlayer::move ( int x, int y )  
{  
    CheckTheHealth();  
    // MACRO 적용.  
    AHNHS_PROTECT_FUNCTION  
    KillMonster( x, y );  
    return 1;  
}  
  
int CPlayer::move ( int x, int y )  
{  
    CheckTheHealth();  
    int nRet =0;  
    // MACRO 적용.  
    AHNHS_PROTECT_FUNCTIONEX (nRet)  
    // 디버깅 용도로 nRet 값을 사용  
    KillMonster( x, y );  
    return 1;
```

```

}

■ 콜백 코드 AHNHS_ACTAPC_DETECT_ABNORMAL_FUNCTION_CALL 추가

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

    switch ( lCode )
    {
        case AHNHS_ACTAPC_DETECT_ABNORMAL_FUNCTION_CALL:
            ExitGame();
            break;
    }
}

```

### 참고

매크로를 적용하려면 적용하려는 프로젝트에 헤더 파일(HShield.h)과 라이브러리(HShield.lib, winmm.lib, version.lib)를 링크하십시오.

보호하려는 함수를 호출하는 게임 모듈에는 반드시 디지털 서명을 하십시오. 단, 보호하려는 함수가 동일 모듈에서 호출되거나 보호하려는 함수가 EXE 모듈에서 호출되는 경우는 예외입니다.

### 2.3.11\_AhnHS\_ThreadStart

HackShield가 적용된 게임 모듈의 내부 함수가 악의적으로 수행되지 않도록 보호하는 함수입니다.

보호가 필요한 게임 모듈의 함수가 단 하나의 스레드에서만 호출되는 경우 사용이 가능하며, 스레드 시작 시점에 한 번만 적용이 가능합니다. HackShield가 시작된 이후, 즉 \_AhnHS\_StartService()가 호출된 이후에 호출해야 하며 \_AhnHS\_StartService()의 호출이 실패하면 해당 함수는 동작하지 않습니다.

NT 이상에서만 동작하며 Win98은 지원하지 않습니다.

```

int
__stdcall
_AhnHS_ThreadStart();

```

#### Parameters

N/A

#### Return value

**HS\_ERR\_OK (Value = 0x000)**

- 설명: 함수 호출 성공

- 원인: 정상적인 경우
- 확인사항: N/A

#### **HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)**

- 설명: 초기화 실패
- 원인: \_AhnHS\_StartService()가 호출되지 않았거나 내부적으로 초기화 작업에 실패한 경우
- 확인사항: \_AhnHS\_StartService()가 호출됐는지 확인

#### **HS\_ERR\_ALREADY\_INITIALIZED (Value = 0x104)**

- 설명: HackShield 이미 초기화
- 원인: \_AhnHS\_ThreadPoolStart()가 이미 초기화된 경우
- 확인사항: \_AhnHS\_ThreadPoolStart()와 \_AhnHS\_ThreadPoolStartEx()가 이미 호출됐는지 확인.  
 \_AhnHS\_ThreadPoolStart()는 보호함수를 호출하는 스레드에서 한 번만 호출해야 하며,  
 \_AhnHS\_ThreadPoolStartEx()와 함께 사용할 수 없음

### **Remarks**

- 옵션 처리
  - \_AhnHS\_Initialize()에서 AHNHS\_CHKOPT\_ABNORMAL\_FUNCTION\_CALL\_V2 옵션 추가
 

```
nRet = _AhnHS_Initialize (      szFullPath,
                                         HS_CallbackProc,           // 콜백 함수
                                         1000,                      // 게임 코드
                                         "B228F291B7D7FAD361D7A4B7", // 라이선스 키
                                         AHNHS_CHKOPT_ALL|AHNHS_CHKOPT_ABNORMAL_FUNCTION_CALL_V2, // 옵션 플래그
                                         AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL);
```
- 보호 함수를 호출하는 스레드에 추가
  - 보호하고자 하는 함수를 호출하는 보호 스레드의 시작 시점에 다음과 같이 적용
  - 보호 스레드가 하나인 경우 적용 가능하므로 \_AhnHS\_ThreadPoolStart()는 한 번만 호출해야 함

```
// 보호 스레드
unsigned __stdcall ThreadProc( void* pParam )
{
    nRet = _AhnHS_ThreadPoolStart();
    if( nRet == HS_ERR_OK )
    {
        while ( TRUE )
        {
            ...
        }
        nRet = _AhnHS_ThreadPoolStop();
    }
    _endthreadex( 0 );
}
```

```
        return 0;  
    }
```

#### ■ 콜백 코드 추가

- AHNHS\_ACTAPC\_DETECT\_ABNORMAL\_FUNCTION\_CALL 추가

```
int __stdcall HS_CallbackProc ( long lCode, long lParamSize, void*  
pParam )  
{  
    TCHAR szMsg[MAX_PATH];  
  
    switch ( lCode )  
    {  
        case AHNHS_ACTAPC_DETECT_ABNORMAL_FUNCTION_CALL:  
            ExitGame();  
            break;  
    }  
}
```

---

#### ⚠ 주의

HackShield의 \_AhnHS\_StartService()와 \_AhnHS\_ThreadStart()가 정상적으로 호출된 이후에 기능이 동작하며, \_AhnHS\_ThreadStartEx()와 함께 사용할 수 없습니다.

---

---

#### ⚠ 주의

HackShield가 적용된 게임 모듈은 반드시 디지털 서명을 해야 합니다.

---

---

#### 🌐 참고

HackShield가 적용된 게임 모듈의 내부 함수가 악의적으로 수행되지 않도록 보호하는 기능에서는 \_AhnHS\_ThreadStart(), \_AhnHS\_ThreadStop(), \_AhnHS\_ThreadStartEx(), \_AhnHS\_ThreadStopEx(), \_AhnHS\_SetProtectedFunction(), \_AhnHS\_CheckProtectedStatus()를 사용합니다.

---

### 2.3.12\_AhnHS\_ThreadStop

HackShield가 적용된 게임 모듈의 내부 함수가 악의적으로 수행되지 않도록 보호하는 함수입니다. \_AhnHS\_ThreadStart()가 적용된 스레드의 종료 시점에 한 번 적용이 가능합니다. 이 함수를 호출해도 \_AhnHS\_ThreadStart()는 다시 호출할 수 없습니다.

```
int  
__stdcall  
_AhnHS_ThreadStop();
```

#### Parameters

N/A

## Return Value

### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

### HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)

- 설명: 초기화 실패.
- 원인: HackShield 를 초기화하지 않았거나 해당 기능을 초기화하지 않은 경우
- 확인사항
  - HackShield 초기화가 정상적으로 수행됐는지 확인
  - \_AhnHS\_ThreadStart()가 호출됐는지 확인

## Remarks

- 기능 적용
  - 스레드 종료 직전에 호출
- 보호 함수를 호출하는 스레드에 추가
  - 보호 함수를 호출하는 스레드의 종료 직전에 다음과 같이 적용

```
// 보호 스레드
unsigned __stdcall ThreadProc( void* pParam )
{
    nRet = _AhnHS_ThreadStart();
    if( nRet == HS_ERR_OK )
    {
        while ( TRUE )
        {
            ...
        }
        nRet = _AhnHS_ThreadStop();
    }
    _endthreadex( 0 );
    return 0;
}
```



### 주의

HackShield의 \_AhnHS\_ThreadStart()를 정상적으로 호출한 후에 호출해야 합니다.

### 2.3.13\_AhnHS\_ThreadStartEx()

HackShield가 적용된 게임 모듈의 내부 함수가 악의적으로 수행되지 않도록 보호하는 함수입니다. 보호가 필요한 게임 모듈의 함수가 하나 이상의 스레드에서 호출되는 경우 사용합니다.

용할 수 있습니다. 각 스레드의 시작 시점에 한 번만 적용할 수 있습니다.  
HackShield가 시작된 이후, 즉 `_AhnHS_StartService()`를 호출한 이후에 호출해야 하며 이  
함수가 실패하면 기능이 동작하지 않습니다.  
NT 이상에서만 동작하며 Win98은 지원하지 않습니다.

```
int
__stdcall
_AhnHS_ThreadStartEx();
```

## Parameters

N/A

## Return Value

### **HS\_ERR\_OK (Value = 0x000)**

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

### **HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)**

- 설명: 초기화 실패
- 원인: `_AhnHS_StartService()`가 호출되지 않았거나 내부적으로 초기화 작업에 실패한 경우
- 확인사항: `_AhnHS_StartService()`가 호출됐는지 확인

### **HS\_ERR\_ALREADY\_INITIALIZED (Value = 0x104)**

- 설명: HackShield 이미 초기화
- 원인: 이 기능이 이미 초기화된 경우
- 확인사항: `_AhnHS_ThreadStart()`를 이미 호출했는지 확인. `_AhnHS_ThreadStartEx()`는  
`_AhnHS_ThreadStart()`와 함께 사용할 수 없음

## Remarks

- 옵션 처리
    - `_AhnHS_Initialize()`에서 AHNHS\_CHKOPT\_ABNORMAL\_FUNCTION\_CALL\_V2 옵션 추가
- ```
nRet = _AhnHS_Initialize (    szFullPath,
                            HS_CallbackProc,           // 콜백함수
                            1000,                      // 게임 코드
                            "B228F291B7D7FAD361D7A4B7", // 라이선스 키
                            AHNHS_CHKOPT_ALL|AHNHS_CHKOPT_ABNORMAL_FUNCTION_CALL_V2, // 옵션 플래그
                            AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL);
```

## ■ 보호 함수를 호출하는 스레드에 추가

- 보호하고자 하는 함수를 호출하는 보호 스레드의 시작 시점에 다음과 같이 적용

```
// 보호 스레드
unsigned __stdcall ThreadProc( void* pParam )
{
    nRet = _AhnHS_ThreadStartEx();
    if( nRet == HS_ERR_OK )
    {
        while ( TRUE )
        {
            ...
        }
        nRet = _AhnHS_ThreadStopEx();
    }
    _endthreadex( 0 );
    return 0;
}
```

## ■ 콜백 코드 추가

- AHNHS\_ACTAPC\_DETECT\_ABNORMAL\_FUNCTION\_CALL 추가

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

    switch ( lCode )
    {
        case AHNHS_ACTAPC_DETECT_ABNORMAL_FUNCTION_CALL:
            ExitGame();
            break;
    }
}
```

### ⚠ 주의

HackShield의 \_AhnHS\_StartService()와 \_AhnHS\_ThreadStartEx()가 정상적으로 호출된 이후에 기능이 동작하며, \_AhnHS\_ThreadStart()가 정상적으로 호출된 이후에 기능이 동작합니다. 보호 스레드가 하나 이상인 경우 적용이 가능하며 각 스레드마다 \_AhnHS\_ThreadStartEx()는 반드시 한 번만 호출돼야 합니다.

### ⚠ 주의

HackShield가 적용된 게임 모듈은 반드시 디지털 서명을 해야 합니다.

### ➊ 참고

HackShield가 적용된 게임 모듈의 내부 함수가 악의적으로 수행되지 않도록 보호하는 기능에서는

---

`_AhnHS_ThreadStart()`, `_AhnHS_ThreadStop()`, `_AhnHS_ThreadStartEx()`, `_AhnHS_ThreadStopEx()`,  
`_AhnHS_SetProtectedFunction()`, `_AhnHS_CheckProtectedStatus()`를 사용합니다.

---

### 2.3.14\_AhnHS\_ThreadStopEx

HackShield가 적용된 게임 모듈의 내부 함수가 악의적으로 수행되지 않도록 보호하는 함수입니다. `_AhnHS_ThreadStartEx()`가 적용된 스레드의 종료 시점에 한 번 적용이 가능합니다.

```
int  
__stdcall  
_AhnHS_ThreadStopEx () ;
```

#### Parameters

N/A

#### Return Value

##### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

##### HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)

- 설명: 초기화 실패.
- 원인: HackShield 를 초기화하지 않았거나 해당 기능이 초기화되지 않은 경우
- 확인사항
  - HackShield 가 초기화가 정상적으로 수행됐는지 확인
  - `_AhnHS_ThreadStartEx()`를 호출했는지 확인

#### Remarks

- 기능 적용
  - 스레드 종료 직전에 호출
- 보호 함수를 호출하는 보호 스레드에 추가
  - 보호 함수를 호출하는 스레드의 종료 직전에 다음과 같이 적용

```
// 보호 스레드  
unsigned __stdcall ThreadProc( void* pParam )  
{  
    nRet = _AhnHS_ThreadStartEx();
```

```

    if( nRet == HS_ERR_OK )
    {
        while ( TRUE )
        {
            ...
        }
        nRet = _AhnHS_ThreadStopEx();
    }
    _endthreadex( 0 );
    return 0;
}

```

### ⚠ 주의

HackShield의 \_AhnHS\_ThreadStartEx()가 정상적으로 호출된 이후에 호출해야 합니다.

## 2.3.15\_AhnHS\_SetProtectedFunction

HackShield가 적용된 게임 모듈의 내부 함수가 악의적으로 수행되지 않도록 보호하는 함수입니다. 보호 함수 내부에 적용하여 보호 함수 호출 시 함께 호출될 수 있도록 합니다. 보호 함수는 식별 가능하도록 함수 단위로 인덱싱하고, 해당 인덱스 정보를 파라미터로 넘겨줍니다. \_AhnHS\_ThreadStart() 또는 \_AhnHS\_ThreadStartEx()가 호출된 이후에 호출해야 합니다.

```

int
__stdcall
_AhnHS_SetProtectedFunction (IN DWORD dwIndex);

```

### Parameters

| Parameter | Value | Description                           |
|-----------|-------|---------------------------------------|
| dwIndex   | DWORD | 보호 함수 식별을 위한 인덱스<br>보호 함수 관리 등에 사용 가능 |

### Return Value

#### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

#### HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)

- 설명: 초기화 실패

- 원인: HackShield 를 초기화하지 않았거나 해당 기능이 초기화되지 않은 경우
- 확인사항
  - HackShield 가 초기화가 정상적으로 수행됐는지 확인
  - \_AhnHS\_ThreadStart()나 \_AhnHS\_ThreadStartEx()를 호출했는지 확인

#### **HS\_ERR\_ALREADY\_EXISTED (Value = 0x303)**

- 설명: 보호 대상 함수가 이미 호출
- 원인: 정상적인 경우
- 확인사항: N/A

### **Remarks**

- 보호 함수에 추가
  - 보호 함수에 대한 인덱싱 작업이 필요하며 보호 함수 내부에 다음과 같이 적용

```
// 보호 함수를 식별하기 위해 인덱싱이 필요합니다.

#define FUNC_IDX_SETCOUNT 0x01
int g_nCount;

// 보호 대상 함수
BOOL SetCount ()
{
    g_nCount++;
    ...
    // 보호 함수가 리턴되기 직전에 호출합니다.
    _AhnHS_SetProtectedFunction (FUNC_IDX_SETCOUNT);
    return TRUE;
}

int GetCount ()
{
    .....
    return g_nCount;
}

// 보호 스레드에서 보호 함수 호출
unsigned __stdcall ThreadProc( void* pParam )
{
    int nCount;

    nRet = _AhnHS_ThreadStart();
    // 또는 nRet = _AhnHS_ThreadStartEx();

    if( nRet == HS_ERR_OK )
    {
```

```

nCount = GetCount();

while ( TRUE )
{
    // 함수 인덱스가 FUNC_IDX_SetCount인 보호 함수 호출
    SetCount();
}

nRet = _AhnHS_ThreadStop();
// 또는 nRet = _AhnHS_ThreadStopEx();
}
_endthreadex( 0 );
return 0;
}

```

#### ⚠ 주의

HackShield의 \_AhnHS\_ThreadStart()나 \_AhnHS\_ThreadStartEx()가 정상적으로 호출된 이후에 기능이 정상적으로 동작합니다.

#### ⚠ 주의

\_AhnHS\_SetProtectedFunction()를 적용했을 때 게임 성능에 영향을 주지 않는지 확인하십시오.

### 2.3.16 \_AhnHS\_CheckProtectedStatus()

HackShield가 적용된 게임 모듈의 내부 함수가 악의적으로 수행되지 않도록 보호하는 함수입니다. 보호 함수에 대한 정보를 전달합니다. 게임에서 비정상 함수 호출 기능이 정상적으로 동작 중인지 확인할 수 있습니다.

단, 게임에서는 보호 함수의 호출 카운트 및 보호 스레드 수 등의 관리해야 하며, HackShield에서 전달하는 정보와 비교하여 허용 오차 범위를 벗어나는 경우 게임이 종료될 수 있도록 해야 합니다.

이 함수를 통해 요청 가능한 타입은 다음과 같습니다.

- EAGLE\_AFCEX\_QUERYTYPE\_THREADCOUNT: 보호 스레드 카운트
- EAGLE\_AFCEX\_QUERYTYPE\_FUNCTIONCOUNT: 보호 함수 호출 카운트

요청 가능한 타입을 정의하는 방식은 다음과 같습니다.

```
#define EAGLE_AFCEX_QUERYTYPE_THREADCOUNT      0x1
#define EAGLE_AFCEX_QUERYTYPE_FUNCTIONCOUNT     0x2
```

**그림 13 \_AhnHS\_CheckProtectedStatus()를 통해 요청 가능한 타입**

\_AhnHS\_CheckProtectedStatus()의 파라미터로 전달하는 정보는 AHNHS\_AFCEX\_QUERY\_INFO 구조체 형태이며 멤버 변수는 다음과 같습니다.

```
typedef struct
```

```

{
    DWORD          dwType;
    DWORD          dwIndex;
    int            nCount;
} AHNHS_AFCEX_QUERY_INFO, *PAHNHS_AFCEX_QUERY_INFO;

```

**그림 14 \_AhnHS\_CheckProtectedStatus()의 정보 전달 구조체**

- dwType : 요청 타입. 게임에서 설정하며 EAGLE\_AFCEX\_QUERYTYPE\_THREADCOUNT 또는 EAGLE\_AFCEX\_QUERYTYPE\_FUNCTIONCOUNT 중 선택
- dwIndex : 보호 함수의 인덱스 값. 게임에서 설정하며 dwType 이 EAGLE\_AFCEX\_QUERYTYPE\_FUNCTIONCOUNT 인 경우에만 의미가 있음
- nCount: 게임에서 초기화. HackShield 에서는 요청 타입(dwType)과 인덱스(dwIndex)에 맞는 카운트를 전달

```

int
__stdcall
_AhnHS_CheckProtectedStatus( IN OUT PVOID pData );

```

### Parameters

| Parameter | Value | Description                                     |
|-----------|-------|-------------------------------------------------|
| pData     | PVOID | [IN OUT] 요청 정보<br>AHNHS_AFCEX_QUERY_INFO 구조체 정보 |

### Return Value

#### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

#### HS\_ERR\_INVALID\_PARAM (Value = 0x 002)

- 설명: 유효하지 않은 파라미터 입력
- 원인: 파라미터로 입력된 데이터가 NULL 인 경우
- 확인사항: 게임에서 파라미터를 정상적으로 입력하고 있는지 확인. 개발 과정에서만 발생할 수 있으므로 별도 처리는 하지 않아도 됨

#### HS\_ERR\_NOT\_INITIALIZED (Value = 0x003)

- 설명: 초기화 실패
- 원인: HackShield 를 초기화하지 않았거나 이 기능이 초기화 되지 않은 경우

#### ■ 확인사항

- HackShield 가 초기화가 정상적으로 수행됐는지 확인
- \_AhnHS\_ThreadStart()나 \_AhnHS\_ThreadStartEx()를 정상적으로 호출했는지 확인

#### HS\_ERR\_INVALID\_DATA (Value = 0x304)

- 설명: 유효하지 않은 데이터를 확인
- 원인: 파라미터로 넘겨 준 데이터가 NULL 인 경우
- 확인사항: 게임에서 파라미터를 정상적으로 입력하고 있는지 확인. 개발 과정에서만 발생할 수 있으므로 별도 처리는 하지 않아도 됨

#### HS\_ERR\_NOT\_SUPPORTED (Value = 0x 305)

- 설명: 지원하지 않는 운영체제
- 원인: 게임을 Windows 9X 계열의 운영체제에서 실행한 경우
- 확인사항: Windows NT 이상의 운영체제에서만 지원하는 기능이므로 Windows 9X 계열의 운영체제에서는 이 함수를 이용한 검증 작업을 수행하지 않도록 함

### Remarks

#### ■ 기능 적용

- \_AhnHS\_CheckProtectedStatus()로 게임에서 필요한 정보를 HackShield 에 요청 가능
- 게임에서는 보호 함수(\_AhnHS\_SetProtectedFunction() 적용한 함수)의 호출 카운트를 관리하고 HackShield 로부터 받은 정보와 비교하여 유효성을 검증하는 로직이 필요
- HackShield 에서 전달해 주는 정보와 게임이 관리하고 있는 정보를 비교하여 게임 내부적으로 정의한 오차 범위를 넘어서면 게임을 종료하도록 함

```
DWORD CheckPotectStatus()
{
    DWORD dwRet;
    PAHNHS_AFCEX_QUERY_INFO pstQueryInfo;
    pstQueryInfo =
        (PAHNHS_AFCEX_QUERY_INFO)malloc(sizeof(AHNHS_AFCEX_QUERY_INFO));

    if( NULL != pstQueryInfo )
    {
        // 함수 호출 정보 요청
        pstAfcQueryInfo->dwType =
            EAGLE_AFCEX_QUERYTYPE_FUNCTIONCOUNT;

        pstAfcQueryInfo->dwIndex =
            AFC_PROTECTED_FUNCTION_INDEX_1;

        pstAfcQueryInfo->nCount = 0;
```

```

        nRet =
            _AhnHS_CheckProtectedStatus((PVOID) pstAfcQueryInfo);

        if( nRet == HS_ERR_OK )
        {
            // HackShield에서 요청한 정보와 게임에서 관리하는
            // 정보가 오차 범위를 벗어나면 해킹이 의심되므로
            // 게임을 종료한다.

            int nHackShieldCount= pstAfcQueryInfo->nCount;

        }

        free(pstAfcQueryInfo);
        pstAfcQueryInfo = NULL;
    }

    Return dwRet;
}

```

---

#### **⚠ 주의**

\_AhnHS\_CheckProtectedStatus()로부터 전달 받는 카운트 값은 integer형 데이터입니다. 최대값 이후에는 음수와 양수 값이 반복되므로 게임에서 유효성 검사 시 주의하십시오.

\_AhnHS\_CheckProtectedStatus() 호출 빈도는 게임의 성능에 영향을 줄 수 있으므로 호출 시에 주의하십시오.

---

#### **⚠ 주의**

게임에서 구현되는 유효성 검증 로직은 가상화 처리 등을 이용한 보안이 반드시 필요합니다.

---

### 2.3.17\_AhnHS\_QueryPerformanceCounter()

안전한 QueryPerformanceCounter 값을 얻게 하는 함수입니다.

```

BOOL
__stdcall
_AhnHS_QueryPerformanceCounter(
    OUT LARGE_INTEGER *lpPerformanceCount,
    OUT int *pErr);

```

#### Parameters

| Parameter          | Value         | Description                                 |
|--------------------|---------------|---------------------------------------------|
| lpPerformanceCount | LARGE_INTEGER | 현재 QueryPerformanceCounter 값을 받는 변수에 대한 포인터 |

|      |      |                                                 |
|------|------|-------------------------------------------------|
| pErr | Int* | _AhnHS_QueryPerformanceCounter()<br>실패 시 상세 에러값 |
|------|------|-------------------------------------------------|

두 번째 파라미터인 pErr의 반환값은 다음과 같습니다.

#### **HS\_ERR\_NOT\_INITIALIZED(Value = 0x003)**

- 설명: 초기화 실패
- 원인: HackShield 동작 시 사용하는 시스템 라이브러리가 정상적으로 로드되지 않는 경우
- 확인사항: 지속적인 에러 발생 시 앱 HackShield 기술 지원에 문의

#### **HS\_ERR\_INVALID\_FILES (Value = 0x101)**

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize()를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: HackShield 가 시작되지 않았거나 해킹으로 인해 HackShield 가 정상 동작하지 않음

### **Return Value**

#### **TRUE**

- 설명: 성공

#### **FALSE**

- 설명: 실패
- 확인사항: 두 번째 파라미터 값 확인

### **Remarks**

- \_AhnHS\_QueryPerformanceCounter() 반환값이 FALSE 인 경우, 파라미터 값으로 세부 에러 확인 가능
- 두 번째 파라미터 값이 ERR\_NOT\_INITIALIZED 이거나 ERR\_INVALID\_FILES 가 아니어서 \_AhnHS\_QueryPerformanceCounter() 자체가 실패한 경우

```

int nErr = 0;
LARGE_INTEGER liCurrent = { 0 };

if( FALSE == _AhnHS_QueryPerformanceCounter ( & liCurrent, &nErr ) )
{
    if (nErr == HS_ERR_NOT_INITIALIZED)
        // HackShield 초기화 실패
    else if(nErr == HS_ERR_INVALID_FILES)
        // HackShield 초기화 실패
}

```

```

        else
        {
            // QueryPerformanceCounter API FAILED.
        }
    }
else
{
    // 성공
}

```

### 2.3.18\_AhnHS\_QueryPerformanceFrequency()

안전한 QueryPerformanceFrequency 값을 얻게 하는 함수입니다.

```

BOOL
__stdcall
_AhnHS_QueryPerformanceFrequency (
    OUT LARGE_INTEGER *lpFrequency,
    OUT int *pErr);

```

#### Parameters

| Parameter   | Value         | Description                                    |
|-------------|---------------|------------------------------------------------|
| lpFrequency | LARGE_INTEGER | 현재 QueryPerformanceFrequency 값을 받는 변수에 대한 포인터  |
| pErr        | Int*          | _AhnHS_QueryPerformanceFrequency() 실패 시 상세 에러값 |

두 번째 파라미터인 pErr의 반환값은 다음과 같습니다.

#### HS\_ERR\_NOT\_INITIALIZED(Value = 0x003)

- 설명: 초기화 실패
- 원인: HackShield 동작 시 사용하는 시스템 라이브러리가 정상적으로 로드되지 않은 경우
- 확인사항: 지속적인 에러 발생 시 안랩 HackShield 기술 지원에 문의

#### HS\_ERR\_INVALID\_FILES (Value = 0x101)

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize()를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: HackShield 가 시작되지 않았거나 해킹으로 인해 HackShield 가 정상 동작하지 않음

## Return Value

### TRUE

- 설명: 성공

### FALSE

- 설명: 실패
- 확인사항: 두 번째 파라미터 값 확인

## Remarks

- \_AhnHS\_QueryPerformanceFrequency()의 반환값이 FALSE 인 경우, 파라미터 값으로 세부 에러 확인 가능
- 두 번째 파라미터 값이 ERR\_NOT\_INITIALIZED 이거나 ERR\_INVALID\_FILES 가 아니어서 \_AhnHS\_QueryPerformanceFrequency() 자체가 실패한 경우

```
int nRet = 0;
LARGE_INTEGER liFrequency = { 0 };

if( FALSE == _AhnHS_QueryPerformanceFrequency ( & liFrequency, &nRet ) )
{
    if (nRet == HS_ERR_NOT_INITIALIZED)
        // HackShield 초기화 실패
    else if(nRet == HS_ERR_INVALID_FILES)
        // HackShield 초기화 실패
    else
    {
        // QueryPerformanceFrequency API FAILED.
    }
}
else
{
    // 성공
}
```

## 2.3.19\_AhnHS\_GetTickCount()

안전한 GetTickCount 값을 얻게 하는 함수입니다.

```
unsigned long
__stdcall
_AhnHS_ GetTickCount (OUT int *pErr);
```

## Parameters

| Parameter | Value | Description                       |
|-----------|-------|-----------------------------------|
| pErr      | Int*  | _AhnHS_GetTickCount() 실패 시 상세 에러값 |

pErr의 반환값은 다음과 같습니다.

### HS\_ERR\_NOT\_INITIALIZED(Value = 0x003)

- 설명: 초기화 실패
- 원인: HackShield 동작 시 사용하는 시스템 라이브러리가 정상적으로 로드되지 않은 경우
- 확인사항: 지속적인 에러 발생 시 안랩 HackShield 기술 지원에 문의

### HS\_ERR\_INVALID\_FILES (Value = 0x101)

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize()를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: HackShield 가 시작되지 않았거나 해킹으로 인해 HackShield 가 정상 동작하지 않음

## Return Value

### TRUE

- 설명: 성공

### FALSE

- 설명: 실패
- 확인사항: Parameter 값을 확인

## Remarks

- \_AhnHS\_GetTickCount()의 반환값이 FALSE 인 경우, 파라미터 값으로 세부 에러 확인 가능
- 파라미터 값이 ERR\_NOT\_INITIALIZED 이거나 ERR\_INVALID\_FILES 가 아니어서 \_AhnHS\_GetTickCount() 자체가 실패한 경우

---

```
int nRet = 0;
unsigned long ulTime = _AhnHS_GetTickCount ( &nRet ) )
if ( ulTime == 0 )
{
    if (nRet == HS_ERR_NOT_INITIALIZED)
        // HackShield 초기화 실패
    else if(nRet == HS_ERR_INVALID_FILES)
        // HackShield 초기화 실패
    else
    {
        // GetTickCount API FAILED.
    }
}
else
{
    // 성공
}
```

---

# 3장

## HackShield 업데이트 기능

3.1 HackShield 업데이트 기능 /77

3.2 애플리케이션 프로그래밍 /79

3.3 애플리케이션 프로그래밍 인터페이스 /88

## 3.1 HackShield 업데이트 기능

---

HackShield의 업데이트 기능을 소개하고 이 기능을 구현하기 위한 방법을 설명합니다.

### 3.1.1 기능 소개

HackShield 업데이트 기능은 HackShield 패치 시 고객 패치 기간에 빌드하고 고객 런처를 통해 업데이트하는 기존의 방식을 개선하기 위해 구현됐습니다. HackShield 전용 업데이트로 빠르게 HackShield 모듈/엔진 패치를 할 수 있으며 해킹툴에 신속하게 대응할 수 있습니다.

HackShield 업데이트 기능은 다음과 같습니다.

- HackShield 모듈 업데이트
- 엔진 업데이트
- HackShield 업데이트 ON/OFF
- 스플래쉬 이미지

#### **HackShield 모듈 업데이트**

HackShield 업데이트는 HackShield 패치가 필요한 경우 별도의 작업을 거치지 않고 손쉽게 HackShield 모듈을 업데이트합니다.

#### **엔진 업데이트**

새로운 해킹툴에 빠르게 대처하기 위해 시그니처와 휴리스틱 엔진을 업데이트합니다.

#### **HackShield 업데이트 ON/OFF**

HackShield 업데이트 기능 사용 중 부득이하게 HackShield 업데이트 기능을 사용하지 않아야 하는 경우에 대비하기 위해 패치셋 설정에 따라 HackShield 업데이트 기능을 ON/OFF 할 수 있습니다.

#### **스플래쉬 이미지**

HackShield 업데이트 진행 중 화면 우측 하단과 중앙에 사용자가 임의로 지정한 이미지를 보여줍니다.

### 3.1.2 기능 특징

HackShield 업데이트 기능의 특징은 다음과 같습니다.

## 인터페이스 함수(API)

HSUpChk.lib에서 제공하는 API를 사용하여 게임의 런처나 실행 파일에서 업데이트 관련 함수를 제일 먼저 호출해서 HackShield 모듈 및 엔진을 업데이트합니다.

## 업데이트 환경 파일 생성 프로그램

업데이트 서버 환경 파일을 제공하여 업데이트할 서버를 임의대로 설정할 수 있습니다. 다중 URL을 지원하여 여러 대의 업데이트 서버를 구축할 수도 있습니다.

## 테스트 프로그램

HSUpChk.lib의 API를 사용하여 구현한 테스트용 프로그램인 Amazon.exe를 제공합니다. Amazon.exe는 기존의 HackShield 테스트 기능과 HackShield 업데이트의 테스트 기능을 제공합니다.

### 3.1.3 시스템 구조

HackShield 업데이트는 HSUpChk.lib를 제공하여 클라이언트에서 API를 호출하여 업데이트 할 수 있게 합니다. HackShield 업데이트의 전체 구조 및 동작 원리는 다음과 같습니다.

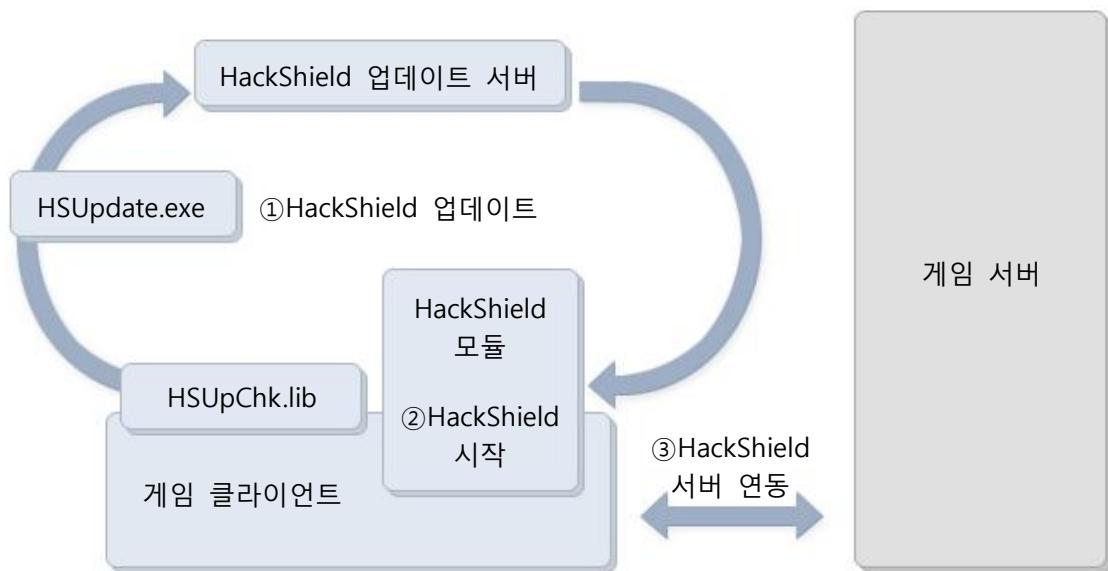


그림 15 HackShield 업데이트 동작 원리

위 그림의 동작 순서에 대한 설명은 다음과 같습니다.

- ①HSUpChk.lib에서 제공하는 `_AhnHS_HSUpdate()`를 호출하여 HackShield 업데이트 서버에 접속을 하고 HackShield 모듈 및 엔진을 업데이트합니다.
- ②HShield.lib에서 제공하는 `_AhnHS_Initialize()`, `_AhnHS_Start()`를 호출하여 HackShield를 실행합니다.
- ③게임 서버에 접속하여 서버 연동을 시작합니다.

위 그림의 주요 항목에 대한 설명은 다음과 같습니다.

### **HSUpChk.lib(HackShield 업데이트 라이브러리)**

게임 클라이언트가 사용합니다. 업데이트 함수가 호출되면 HsUpdate.exe가 호출되어 필요한 HackShield 모듈을 업데이트할 수 있게 합니다.

### **HSUpdate.env(업데이트 환경 파일)**

HSUpSetEnv.exe 툴에서 생성되는 파일입니다. 업데이트 서버 환경 설정 정보를 담고 있습니다.

### **HSUpSetEnv.exe(업데이트 환경 파일 생성 프로그램)**

HSUpdate.env 업데이트 환경 설정 파일을 생성하는 툴입니다. 업데이트 서버의 환경을 설정할 수 있습니다.

### **noupdate.ui(업데이트 비활성화(OFF) 설정 파일)**

noupdate.ui는 업데이트를 일시적으로 비활성화(OFF) 시킬 때 사용합니다. 업데이트를 사용하던 중 부득이하게 업데이트 기능을 비활성화해야 하는 경우, 서비스 중인 업데이트 서버에서 HackShield 패치셋 하위(ahn.ui, autoup.exe 등과 동일한 경로)에 noupdate.ui 파일을 위치시키면 HackShield 업데이트 기능이 비활성화(OFF)됩니다.

업데이트 기능을 정상적으로 활성화(ON) 시키려면 서비스 중인 업데이트 서버에서 HackShield 패치셋 폴더 하위(ahn.ui, autoup.exe 등과 동일한 경로)에 위치시킨 noupdate.ui 파일을 삭제하십시오.

## **3.2 애플리케이션 프로그래밍**

HSUpChk.lib에서 제공하는 API를 사용해서 HackShield의 업데이트 기능을 구현하는 방법을 설명합니다.

---

### **참고**

이 장에서 사용하고 있는 샘플 코드는 Microsoft Visual C++ 6.0을 기준으로 한 C/C++ 언어로 작성했습니다. 프로그래밍에 사용되는 언어는 각 프로그램의 특성 및 시스템 환경에 따라서 변경할 수 있습니다.

---

### **3.2.1 프로그래밍 준비**

HSUpChk.lib를 사용하여 프로그래밍을 시작하기 전에 확인해야 하는 업데이트 관련 파일

목록은 다음과 같습니다.

### 업데이트 관련 파일

업데이트 관련 파일은 다음과 같습니다.

표 7 업데이트 관련 파일

| 파일 이름        | 설치 폴더          | 설명              |
|--------------|----------------|-----------------|
| HSUpChk.lib  | [프로그램 소스 폴더]   | 서버에서 사용할 헤더 파일  |
| HSUpChk.h    | [프로그램 소스 폴더]   | 서버에서 사용할 DLL 파일 |
| AhnUpCtl.dll | [Game]\HShield | 업데이트 DLL        |
| AhnUpGS.dll  | [Game]\HShield | 업데이트 DLL        |
| HSInst.dll   | [Game]\HShield | UI DLL          |
| HSUpdate.env | [Game]\HShield | 업데이트 환경 설정 파일   |
| HSUpdate.exe | [Game]\HShield | 업데이트 실행 파일      |
| V3Hunt.dll   | [Game]\HShield | 업데이트 DLL        |
| V3InetGS.dll | [Game]\HShield | 업데이트 DLL        |

고객의 HackShield 자동 업데이트 서버에 필요한 파일 이름과 구성은 다음과 같습니다.  
PatchSet의 경로는 [HackShield SDK] \PatchSet입니다.

표 8 업데이트 PatchSet

| 폴더 구조    |       |    |               |
|----------|-------|----|---------------|
| PatchSet | patch | 39 | ahn.id        |
|          |       |    | ahn.ui        |
|          |       |    | ahn.uic-      |
|          |       |    | ahn rpt.ex-   |
|          |       |    | ahn rpt.in-   |
|          |       |    | ahn upctl.dl- |
|          |       |    | ahn upgs.dl-  |
|          |       |    | Bldinfo.in-   |
|          |       |    | ehsvc.dl-     |
|          |       |    | Hshield.da-   |
|          |       |    | hsinst.dl-    |
|          |       |    | Hslogmgr.ex-  |
|          |       |    | hsupdate.ex-  |
|          |       |    | v3hunt.dl-    |
|          |       |    | v3inetgs.dl-  |
|          | win   | e  | b             |
|          |       |    | b_echo_si     |
|          |       |    | asc_com.dl    |
|          |       |    | asc_dh.dl     |
|          |       |    | asc_fse.dl    |
|          |       |    | asc_intg.dl   |
|          |       |    | asc_mmgr.dl   |
|          |       |    | asc_unp.dl    |

|                                                                                                                                                                                                                                                                           |              |            |             |         |          |        |         |           |              |            |            |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|------------|-------------|---------|----------|--------|---------|-----------|--------------|------------|------------|
|                                                                                                                                                                                                                                                                           |              |            | fse_base.dl |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | fse_fact.dl |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | fse_pe.dl   |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | gfs_base.dl |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | gfs_fact.dl |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | gfs_file.dl |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | gfs_mem.dl  |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | gfs_os.dl   |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | gfs_proc.dl |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | gfs_util.dl |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           | b_sign_hs    |            | 0asc.sc     |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | 0scsure.sc  |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | 0sgame.sc   |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | 0spe3f.sc   |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | moduler.sc  |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              |            | option.sc   |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           |              | V3_echo_hs | V3pro32s.dl |         |          |        |         |           |              |            |            |
|                                                                                                                                                                                                                                                                           | hs_heuristic | 3n.mh      |             |         |          |        |         |           |              |            |            |
| <hr/>                                                                                                                                                                                                                                                                     |              |            |             |         |          |        |         |           |              |            |            |
| <table border="1"> <tbody> <tr><td>ahn.ahc</td></tr> <tr><td>ahni2.id</td></tr> <tr><td>Ahn.ui</td></tr> <tr><td>Ahn.uic</td></tr> <tr><td>Ahni2.dll</td></tr> <tr><td>ahnupctl.dll</td></tr> <tr><td>autoup.exe</td></tr> <tr><td>v3bz32.dll</td></tr> </tbody> </table> |              |            |             | ahn.ahc | ahni2.id | Ahn.ui | Ahn.uic | Ahni2.dll | ahnupctl.dll | autoup.exe | v3bz32.dll |
| ahn.ahc                                                                                                                                                                                                                                                                   |              |            |             |         |          |        |         |           |              |            |            |
| ahni2.id                                                                                                                                                                                                                                                                  |              |            |             |         |          |        |         |           |              |            |            |
| Ahn.ui                                                                                                                                                                                                                                                                    |              |            |             |         |          |        |         |           |              |            |            |
| Ahn.uic                                                                                                                                                                                                                                                                   |              |            |             |         |          |        |         |           |              |            |            |
| Ahni2.dll                                                                                                                                                                                                                                                                 |              |            |             |         |          |        |         |           |              |            |            |
| ahnupctl.dll                                                                                                                                                                                                                                                              |              |            |             |         |          |        |         |           |              |            |            |
| autoup.exe                                                                                                                                                                                                                                                                |              |            |             |         |          |        |         |           |              |            |            |
| v3bz32.dll                                                                                                                                                                                                                                                                |              |            |             |         |          |        |         |           |              |            |            |

PatchSet의 내부 구조에 대한 파일 및 폴더는 다음과 같습니다.

**표 9 업데이트 서버에 설치되어야 하는 파일 목록**

| 파일/ 폴더 이름             | 설명                          |
|-----------------------|-----------------------------|
| ahn.ui                | 업데이트 정보 파일(엔진 파일 버전 정보)     |
| ahn.uic               | 업데이트 정보 파일(엔진 파일 버전 정보)     |
| ahni2.dll             | 업데이트 파일                     |
| ahnupctl.dll          | 업데이트 파일                     |
| autoup.exe            | 업데이트 파일                     |
| v3bz32.dll            | 업데이트 파일                     |
| patch\39\3n.mh-       | 휴리스틱 엔진 파일(압축된 파일)          |
| patch\39\ahn.ui       | 업데이트 정보 파일(패치 파일 버전 정보)     |
| patch\39\ahn.uic      | 업데이트 정보 파일(패치 파일 버전 정보)     |
| patch\39\ahnupctl.dl- | 업데이트 파일(압축 파일)              |
| patch\39\ahnupgs.dl-  | 업데이트 파일(압축 파일)              |
| patch\39\ehsvc.dl-    | HackShield 인터페이스 DLL(압축 파일) |
| patch\39\hshield.da-  | HackShield 관련 dat 파일(압축 파일) |

|                                |                             |
|--------------------------------|-----------------------------|
| patch\39\hsinst.dl-            | 업데이트 파일(압축 파일)              |
| patch\39\hsupdate.ex-          | 업데이트 실행 파일(압축 파일)           |
| patch\39\v3hunt.dl-            | 업데이트 파일(압축 파일)              |
| patch\39\v3inetgs.dl           | 업데이트 파일(압축 파일)              |
| win\e\b\b_echo_sl\asc_com.dl   | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\asc_dh.dl    | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\asc_fse.dl   | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\asc_intg.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\asc_mmgr.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\asc_unp.dl   | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\fse_base.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\fse_fact.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\fse_pe.dl    | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\gfs_base.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\gfs_fact.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\gfs_file.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\gfs_mem.dl   | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\gfs_os.dl    | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\gfs_proc.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_echo_sl\gfs_util.dl  | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |
| win\e\b\b_sign_hs\0asc.sc      | 해킹 툴 패턴 엔진 파일(압축 파일)        |
| win\e\b\b_sign_hs\0scure.sc    | 해킹 툴 패턴 엔진 파일(압축 파일)        |
| win\e\b\b_sign_hs\0sgame.sc    | 해킹 툴 패턴 엔진 파일(압축 파일)        |
| win\e\b\b_sign_hs\0specf3.sc   | 해킹 툴 패턴 엔진 파일(압축 파일)        |
| win\e\b\b_sign_hs\moduler.sc   | 해킹 툴 패턴 엔진 파일(압축 파일)        |
| win\e\b\b_sign_hs\option.sc    | 해킹 툴 패턴 엔진 파일(압축 파일)        |
| win\e\b\v3_echo_hs\v3pro32s.dl | 해킹 툴 탐지 엔진 인터페이스 DLL(압축 파일) |

업데이트에 사용되는 데이터 파일은 다음과 같습니다. 데이터 파일 경로는 HackShield SDK] # Data#입니다.

**표 10** 업데이트에 사용되는 데이터 파일

| 파일 이름       | 설명                                      |
|-------------|-----------------------------------------|
| noupdate.ui | 업데이트 기능을 일시적으로 비활성화(OFF) 시킬 때 필요한 설정 파일 |

업데이트 이후 자동으로 생성되는 파일은 다음과 같습니다.

**표 11** HackShield 업데이트 이후 자동 생성 파일

| 파일 이름        | 설치 폴더                   | 설명                   |
|--------------|-------------------------|----------------------|
| ahn.ui       | [HackShield 폴더]/Update/ | 업데이트 정보 파일           |
| ahn.uic      | [HackShield 폴더]/Update/ | 업데이트 정보 파일           |
| ahni2.dll    | [HackShield 폴더]/Update/ | 업데이트 DLL             |
| ahnupctl.dll | [HackShield 폴더]/Update/ | 업데이트 DLL             |
| autoup.exe   | [HackShield 폴더]/Update/ | 업데이트 실행 파일(파일 패치 처리) |

|            |                         |                    |
|------------|-------------------------|--------------------|
| v3bz32.dll | [HackShield 풀더]/Update/ | 업데이트 DLL(파일 압축 처리) |
|------------|-------------------------|--------------------|

### 3.2.2 프로그래밍 적용

프로그래밍을 하기 위해 서버와 클라이언트를 준비하고 적용하는 방법은 다음과 같습니다.

#### 서버 적용 방법

HackShield 업데이트 서버는 고객이 준비하여 설정하고 운영해야 합니다. HackShield 업데이트 서버의 권장 사양 및 최소 사양은 다음과 같습니다.

**표 12 HackShield 업데이트 서버 사양**

| 항목    | 권장 사양               | 최소 사양                |
|-------|---------------------|----------------------|
| 운영체제  | Windows Server 2003 | Windows Server 2000  |
| CPU   | Intel Xeon 3.2 Dual | Intel Pentium IV 2.4 |
| RAM   | 2GB 이상              | 1GB 이상               |
| 하드디스크 | 40GB 이상             | 40GB 이상              |

HackShield 업데이트 서버를 구성하고 설정하는 순서는 다음과 같습니다.

1. IIS 설정 등으로 FTP나 HTTP로 외부에서 접근 가능한 업데이트 서버를 구성합니다.

 참고

FTP의 경우, 개별적인 계정을 설정하거나 anonymous 접근이 가능하도록 설정합니다. 또한 passive 모드로 접속 가능하도록 설정해야 합니다. passive 모드 접속 여부를 확인하려면 Internet Explorer에서 FTP 접근 후 파일 목록이 보이는지 확인하십시오.

2. [HackShield SDK]\PatchSet 풀더에 있는 전체 파일을 서버에 업로드합니다.

 참고

전체 파일 목록은 **표 9 업데이트 서버에 설치되어야 하는 파일 목록**을 참고하십시오.

 주의

PatchSet은 서버의 root 폴더에 하위 폴더를 생성하여 업로드하십시오.

 주의

업데이트 주소를 입력할 때 ahnlab이라는 문자는 사용하지 마십시오. 도메인 주소란에 ahnlab이 포함되어 있으면 접속을 제한하는 특정 웜바이러스가 리포트 된 적이 있습니다. 예를 들어, 업데이트 주소를 ahnlabGame.co.kr/real/로 표시한 경우 주소란에 ahnlab이 있기 때문에 해당 HTTP 사이트에 접속할 수 없습니다.

HackShield 업데이트 서버에서 확장자가 dll이거나 exe인 실행 파일을 다운로드할 수 없는 경우가 있습니다. 이는 IIS 웹 서버의 디렉터리별로 실행 권한이 **스크립트 및 실행 파일**로

설정되어 있는 경우, 실행 파일이 서버에서 실행되어 파일을 다운로드할 수 없기 때문입니다.

이러한 경우에는 IIS(Internet Information Services) 설정을 변경하십시오. 변경 순서는 다음과 같습니다.

1. 제어판->IIS(Internet Information Services) 관리 툴을 선택합니다.
2. 해당 웹 사이트에서 오른쪽 마우스 버튼을 눌러서 속성을 선택합니다.
3. 홈 디렉터리 탭으로 이동합니다.
4. 실행 권한을 없음으로 설정합니다.

## 클라이언트 적용 방법

클라이언트를 구성하고 설정하는 순서는 다음과 같습니다.

1. HSUpChk.lib 를 작업할 프로젝트에 포함시킵니다.
2. 제공되는 HSUpChk.h 를 작업할 소스 파일에 포함시킵니다.

### 참고

HackShield 업데이트 관련 두 개의 API 중 하나를 선택해서 적용하면 됩니다.

\_AhnHS\_HSUpdate(): HackShield 기본 업데이트

\_AhnHS\_HSUpdateEx(): HackShield 기본 업데이트 + env 파일 교체 방지 + 호스트 파일 체크

3. 업데이트 관련 함수인 \_AhnHS\_HSUpdateEx() 나 \_AhnHS\_HSUpdate() 를 호출합니다.

```
// _AhnHS_HSUpdateEx() 호출 예제

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

// HackShield 폴더 위치를 지정합니다.
_tcscpy (szFullPath, _T( ".\\HShield" ) );

AHNHS_EXT_ERRORINFO HsExtError = {0,};

// Monitoring 정보를 지정합니다.
HsExtError.szServer = "127.0.0.1"           // 모니터링주소
HsExtError.szUserId = "Test"                 // 게임 유저 ID
HsExtError.szGameVersion = "3.0.0.1"         // 게임 버전

// _AhnHS_HSUpdateEx() 호출
dwRet = _AhnHS_HSUpdateEx(      szFullPath,           // HackShield 폴더 경로
                           1000 * 600,           // 업데이트 전체 타임 아웃
                           1234,                // 게임 코드
                           AHNHSUPDATE_CHKOPT_HOSTFILE |
                           AHNHSUPDATE_CHKOPT_GAMECODE,
                           HsExtError,
```

```

        1000* 20 );           // 서버 연결 타임아웃
// Ex 함수를 사용하실 때는 반드시 HSUpSetEnv.exe 설정 툴로 env 파일에
// 게임 코드를 입력해야 합니다.
if ( dwRet != ERROR_SUCCESS)
{
    // 에러 처리
    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;
        ...
    }
}

```

```

// _AhnHS_HSUpdate() 호출 예제

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

// HackShield 폴더 위치를 지정합니다.
_tcscpy (szFullPath, _T( ".\\HShield" ) );

// _AhnHS_HSUpdate() 호출
dwRet = _AhnHS_HSUpdate (      szFullPath,          // HackShield 폴더 경로
                            1000 * 600,          // 업데이트 전체 타임아웃
                            1000* 20 );          // 서버 연결 타임아웃

if ( dwRet != ERROR_SUCCESS)
{
    // 에러 처리
    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;
    ...
}
}

```

- 4. 배포할 클라이언트의 [Game Directory]\HShield 폴더에 [HackShield SDK] \Bin\Win\x86\Update 폴더의 전체 파일을 복사해 넣습니다. 복사 후 폴더 파일 구조는 다음과 같습니다.**

```

[Game Directory]\HShield\
3n.mhe
AhnRtp.exe
ahnrp.ini
AhnUpCtl.dll
AhnUpGS.dll
BldInfo.ini
EHSvc.dll
hshield.dat
HSInst.dll
HSLogMgr.exe
HSUpdate.env
HSUpdate.exe
V3Hunt.dll
V3InetGS.dll
v3pro32s.dll

[Game Directory]\HShield\asc
0asc.scd
0scecure.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
```

- [HackShield SDK]\Bin\Win\x86\Util\HSUpSetEnv.exe 파일을 실행합니다.

---

**⚠ 주의**

HSUpSetEnv.exe는 업데이트 설정 파일을 생성하는 툴이므로 배포하지 마십시오.

---

- HSUpdate.env 파일을 생성합니다.

---

**➊ 참고**

자세한 내용은 [9.2 HSUpSetEnv 툴](#)을 참고하십시오.

---

- HSUpdate.env 파일을 [GameDirectory]\HShield 폴더로 복사합니다.

---

**⚠ 주의**

업데이트가 정상적으로 실행되려면 \_AhnHS\_HSUpdate()에 첫번째로 전달되는 파라미터([Game Directory]\HShield)에 [HackShield SDK]\Bin\Win\x86\Update에 있는 업데이트 모듈이 모두 있어야 합니다.

---

- 스플래쉬 이미지를 사용하려면 사용할 이미지 파일 이름을 splash.jpg로 변경한 다음 [Game Directory]\HShield 폴더에 복사합니다.
- 업데이트 시 화면 오른쪽 아래의 HackShield 업데이트 이미지를 변경하려면 사용할 이미지 파일 이름을 hsupdate.jpg로 변경한 다음 [Game Directory]\HShield 폴더에 복사합니다.



---

### !

#### 참고

업데이트 이미지에 대한 참고 사항은 다음과 같습니다.

파일 이름: hsupdate.jpg (대소문자 구분하지 않음)

크기: 200 \* 149

복사할 경로: [Game Directory]\WHShield

설명: HackShield 폴더 내에 hsupdate.jpg가 존재한다면 해당 그림 파일을 로드하여 업데이트 이미지로 사용합니다. 존재하지 않는다면 기본 이미지를 사용합니다.

기본 이미지 사용 시 이미지 하단에 프로그레스 바가 생깁니다.

---

---

### !

#### 참고

스플래쉬 이미지 참고 사항은 다음과 같습니다.

파일 이름: splash.jpg (대소문자 구분하지 않음)

크기: 290 \* 190

복사할 경로: [Game Directory]\WHShield

설명: HackShield 폴더 내에 splash.jpg가 존재한다면 해당 그림 파일을 로드하여 화면 정 가운데에 스플래쉬 이미지로 사용합니다. 존재하지 않는다면 아무 것도 보여주지 않습니다.

---

## 3.3 애플리케이션 프로그래밍 인터페이스

---

HackShield의 업데이트 기능 구현에 필요한 API를 소개합니다.

### 3.3.1 \_AhnHS\_HSUpdateEx()

HackShield 파일을 업데이트하는 함수입니다. HsUpdate.env와 호스트 파일을 체크하는 기능이 있습니다.

```
DWORD __stdcall  
_AhnHS_HSUpdateEx (  
    LPCTSTR szUpdateDir,
```

```

        DWORD           dwTimeOut,
        INT64          i64GameCode,
        DWORD          dwOption,
        AHNHS_EXT_ERRORINFO HsExtErrorInfo,
        DWORD          dwTimeOutPerConnection = 0
    );

```

## Parameters

| Parameter              | Value                   | Description                                                                                                                     |
|------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| szUpdateDir            | LPCTSTR                 | 업데이트 파일이 설치돼 있는 폴더                                                                                                              |
| dwTimeOut              | DWORD<br>(milliseconds) | 업데이트 시 타임 아웃 시간<br>0으로 설정 시 INFINITE으로 설정<br>*네트워크 사정에 따라 600000(10<br>분) 내외의 값으로 설정하는 것을 권<br>장하며 가급적이면 0으로는 설정하<br>지 않는 것을 권장 |
| i64GameCode            | INT64                   | HSUpSetEnv.exe 툴로<br>HSUpdate.env 파일에 게임 코드를<br>저장한 값을 입력<br>(모니터링 기능 사용 시 반드시 입력)                                              |
| dwOption               | DWORD                   | 업데이트 추가 기능 옵션<br>AHNHSUPDATE_CHKOPT_HOSTFILE<br>AHNHSUPDATE_CHKOPT_GAMECODE                                                     |
| HsExtErrorInfo         | AHNHS_EXT_ERRORINFO     | 서버 URL 주소, User ID, Game<br>Version 정보를 가지고 있는 구조체                                                                              |
| dwTimeOutPerConnection | DWORD<br>(milliseconds) | 서버 연결 시 타임 아웃 시간<br>*네트워크 사정에 따라 적절한 값으<br>로 설정<br>*0으로 설정 시, 서버 연결할 때 타<br>임 아웃 기능이 동작하지 않음                                    |

각 파라미터에 대한 자세한 설명은 다음과 같습니다.

■ szUpdateDir [in]

- 배포되는 HackShield 관련 파일들이 존재하는 절대 경로를 설정하며, 해당 폴더에는 업데이트 모듈이 반드시 전부 있어야 함
- 올바른 절대 경로 설정을 위해 GetModuleFileName() 이용 권장(GetCurrentDirectory()는 원하는 경로를 얻지 못할 수도 있음)

■ dwTimeOut [in]

- 함수 호출 후 대기 시간을 의미
- 대기 시간 동안 응답 없으면 함수 호출 실패로 간주하지만 해외와 같이 네트워크 속도가 느린 경우에는 대기 시간을 10 분 정도로 설정할 것을 권장

- 업데이트 실패 시에는 고객 정책에 따라 게임 실행 여부를 판단해야 함
- i64GameCode [in]
- 업데이트 환경 설정 파일을 임의로 바꾸지 못하도록(HSUpdate.env) 업데이트 모듈과 맞쳐주는 역할
  - 업데이트 환경 파일에 저장한 게임 코드 값을 입력
  - 환경 파일에 저장한 게임 코드 값과 다르거나 환경 파일에 게임 코드 값이 없을 경우 HSERROR\_MISMATCH\_ENVFILE 에러 리턴
  - 모니터링 서버에서 게임 식별 코드로 사용되므로 모니터링 기능을 사용할 경우 반드시 입력
- dwOption [in]
- \_AhnHS\_HSUpdateEx()에서 동작하는 기능을 정의하는 옵션
  - AHNHSUPDATE\_CHKOPT\_HOSTFILE: 호스트 파일을 검사하여 업데이트 URL 서버 리스트 중 하나라도 IP 지정이 되어 있다면 HSERROR\_HOSTFILE\_MODIFICATION 에러를 리턴. 호스트 파일에 업데이트 URL 정보를 입력하여 업데이트를 우회하는 것을 차단
  - AHNHSUPDATE\_CHKOPT\_GAMECODE: HSUpdate.env 파일에 저장된 게임 코드와 파라미터로 전달된 i64GameCode 를 비교하여 다른 경우 HSERROR\_MISMATCH\_ENVFILE 에러를 리턴
- HsExtErrorInfo [in]
- 서버 IP, 유저 계정, 게임 버전 정보를 포함하는 구조체
  - szServer 멤버에는 현재 운영되는 HackShield 모니터링 서버 IP, szUserID 멤버에는 유저 계정 정보, szGameVersion 멤버에는 클라이언트의 버전을 입력
  - HackShield 업데이트에 대하여 모니터링 서비스 기능을 적용할 경우 이 구조체에 정보를 반드시 입력해야 하며, HackShield 업데이트 기능에 대해 모니터링 서비스를 적용하지 않는 경우 해당 파라미터를 ZeroMemory 로 초기화
- dwTimeOutPerConnection [in]
- 서버에 연결 시 대기 시간을 의미
  - 대기 시간 동안 응답이 없으면 현재 접속하려는 서버로의 연결은 실패로 간주
  - HSUpSetEnv.exe 를 이용하여 HSUpdate.env 파일에 한 개 이상의 서버를 설정한 경우, 각 서버에 대해 dwTimeOutPerConnection 값을 동일하게 적용
  - '0'으로 설정 시 서버에 연결할 때 대기 시간을 체크하지 않음

## Return Value

**HACKSHIELD\_ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 업데이트 성공
- 원인: 정상적인 경우
- 확인사항: N/A

**HSERROR\_ENVFILE\_NOTREAD (Value = 0x30000010)**

- 설명: HSUpdate.env 읽기 실패

- 원인: env 환경 파일이 없거나 맞지 않는 버전의 HSUpSetEnv.exe 툴을 사용한 경우
- 확인사항
  - 해당 경로에 env 파일 존재 여부 확인
  - HSUpSetEnv.exe 툴이 sdk에 존재하는 최신 버전인지 확인

#### **HSERROR\_ENVFILE\_NOTREADFOUND (Value = 0x30000011)**

- 설명: HSUpdate.env 가 존재하지 않음
- 원인: env 환경 파일이 존재하지 않거나 접근 권한이 없는 경우
- 확인사항: HShield 폴더에 env 파일의 존재 여부 확인

#### **HSERROR\_UPDATE\_INITIALIZE\_FAILED (Value = 0x3000001C)**

- 설명: 모듈 초기화 실패
- 원인: HSUpdate.exe에서 초기화하는 과정에서 에러가 발생한 경우
- 확인사항: 업데이트 내부 에러일 수 있기 때문에 안랩 HackShield 기술 지원에 문의

#### **HSERROR\_ENVFILE\_NOTWRITE (Value = 0x30000020)**

- 설명: HSUpdate.env 쓰기 실패
- 원인: HSUpdate.env 파일 생성 시 읽기 속성이거나 접근 권한이 없는 경우
- 확인사항
  - env 환경 파일에 접근 권한이 있는지 확인
  - 환경 파일을 삭제하고 새로 생성했을 때도 동일한 문제가 발생하는지 확인

#### **HSERROR\_NETWORK\_CONNECT\_FAIL (Value = 0x30000030)**

- 설명: 해당 서버 연결 실패
- 원인: 업데이트 서버(ftp/http)에 접속할 수 없는 경우
- 확인사항
  - 업데이트 서버가 정상적으로 접근되는지 ftp 툴이나 Internet Explorer로 확인
  - Env 환경 파일에 업데이트 주소 및 ID와 비밀번호가 올바르게 입력됐는지 확인
  - 특정 웹의 경우 보안프로그램 사이트 접속을 제한하기 때문에 백신프로그램으로 검사 및 치료
  - 업데이트 서버 최초 구축 시 구축된 서버에서 확장자가 \*.ui 를 인식 못하는 경우에 업데이트 서버에서 해당 파일을 다운로드하지 못하는 경우가 있으므로 업데이트 서버의 파일 구성 요소에 \*.ui 확장자가 인식 가능한지 확인

#### **HSERROR\_LIB\_NOTEDIT\_REG (Value = 0x30000050)**

- 설명: 네트워크 결과 입력 중 에러
- 원인: N/A
- 확인사항: N/A

#### **HSERROR\_NOTFINDFILE (Value = 0x30000060)**

- 설명: HackShield 업데이트 프로그램 관련 파일 찾기 실패
- 원인: N/A
- 확인사항: N/A

#### **HSERROR\_PROTECT\_LISTLOAD\_FAIL (Value = 0x30000070)**

- 설명: HSUpdate.pt 인증 파일 찾기 실패
- 원인: N/A
- 확인사항: N/A

#### **HSERROR\_HSUPDATE\_TIMEOUT (Value = 0x30000090)**

- 설명: HackShield 업데이트 프로그램 업데이트 전송 시간 초과
- 원인: \_AhnHS\_HSUpdate() 호출 시 설정한 타임 아웃 시간 안에 업데이트되지 않았거나 네트워크 상황이 좋지 않은 경우
- 확인사항: N/A

#### **HSERROR\_HSUPDATE\_MODIFICATION (Value = 0x30000091)**

- 설명: HSUpdate.exe 파일이 변경
- 원인: HSUpdate.exe 파일이 HackShield SDK에서 배포하는 파일이 아닌 경우
- 확인사항: N/A

#### **HSERROR\_STRING\_CONVERSION\_FAILED (Value = 0x300000B0)**

- 설명: 유니코드 스트링 변환 실패
- 원인: 유니코드 지원 API 함수에서 스트링 변환에 실패한 경우
- 확인사항: 인자로 전달된 szUpdateDir 값을 확인

#### **HSERROR\_MISMATCH\_ENVFILE (Value = 0x300000C0)**

- 설명: 게임 클라이언트와 업데이트 환경 설정 파일의 짹이 맞지 않음
- 원인: \_AhnHS\_HSUpdateEx() 호출 시 전달한 i64GameCode 값과 HSUpdate.env 파일의 게임 코드 값과 다르거나 HSUpdate.env 파일에 게임 코드가 저장되어 있지 않은 경우
- 확인사항: 파라미터로 전달된 i64GameCode 와 HSUpdate.env 파일에 저장된 게임 코드가 동일한지 확인. AHNHSUPDATE\_CHKOPT\_GAMECODE 옵션으로 해당 기능 ON/OFF 가능

#### **HSERROR\_HOSTFILE\_MODIFICATION (Value = 0x300000D0)**

- 설명: 호스트 파일에 HackShield 업데이트 URL을 지정함
- 원인: HSUpdate.env 파일에 저장된 서버 주소 리스트 중에 하나라도 호스트 파일에 IP 지정이 되어 있어서 감지한 경우

- 확인사항: 호스트 파일에 HackShield 업데이트 URL이 IP 지정이 되어 있는지 확인.  
AHNHSUPDATE\_CHKOPT\_HOSTFILE 옵션으로 해당 기능 ON/OFF 가능

#### **HSERROR\_AUTOUPDATE\_FAIL (Value = 0x300000E0)**

- 설명: HackShield 자동 업데이트 프로세스 중 문제 발생
- 원인: 업데이트 내부 에러일 수 있기 때문에 안랩 HackShield 기술 지원에 문의
- 확인사항: N/A

#### **HSERROR\_UPDATE\_WIN32\_ERROR (Value = 0x3000FFFF)**

- 설명: HackShield 자동 업데이트 프로세스 중 문제 발생
- 원인: 업데이트 내부 에러일 수 있기 때문에 안랩 HackShield 기술 지원에 문의
- 확인사항: N/A

#### **Remarks**

업데이트 함수는 반드시 초기화 함수 `_AhnHS_Initialize()` 호출 전에 호출해야 합니다. 다른 HackShield 함수와 별도로 게임 프로세스를 실행시키는 런처와 같은 실행 파일에서 호출해서 사용할 수 있습니다.

### **3.3.2 \_AhnHS\_HSUpdate()**

HackShield 파일을 업데이트하는 함수입니다.

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

#### **Parameters**

| Parameter   | Value                   | Description                                                                                                                |
|-------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------|
| szUpdateDir | LPCTSTR                 | 업데이트 파일이 설치돼 있는 폴더                                                                                                         |
| dwTimeOut   | DWORD<br>(milliseconds) | 업데이트 시 타임 아웃 시간<br>0으로 설정 시 INFINITE으로 설정<br>*네트워크 사정에 따라 600000(10분) 내외의<br>값으로 설정하는 것을 권장하며 가급적이면 0으<br>로는 설정하지 않는 것을 권장 |

|                        |                         |                                                                                    |
|------------------------|-------------------------|------------------------------------------------------------------------------------|
| dwTimeOutPerConnection | DWORD<br>(milliseconds) | 서버 연결 시 타임 아웃 시간<br>*네트워크 사정에 따라 적절한 값으로 설정<br>*0으로 설정 시 서버 연결 시 타임 아웃 기능이 동작하지 않음 |
|------------------------|-------------------------|------------------------------------------------------------------------------------|

각 파라미터에 대한 자세한 설명은 다음과 같습니다.

- szUpdateDir [in]
  - 배포되는 HackShield 관련 파일들이 존재하는 절대 경로를 설정하며, 해당 폴더에는 업데이트 모듈이 반드시 전부 있어야 함
  - 올바른 절대 경로 설정을 위해 GetModuleFileName() 이용 권장(GetCurrentDirectory()는 원하는 경로를 얻지 못할 수도 있음)
- dwTimeOut [in]
  - 함수 호출 후 대기 시간을 의미
  - 대기 시간 동안 응답 없으면 함수 호출 실패로 간주하지만 해외와 같이 네트워크 속도가 느린 경우에는 대기 시간을 10 분 정도로 설정할 것을 권장
  - 업데이트 실패 시에는 고객 정책에 따라 게임 실행 여부를 판단해야 함
- dwTimeOutPerConnection [in]
  - 서버에 연결 시 대기 시간을 의미
  - 대기 시간 동안 응답이 없으면 현재 접속하려는 서버로의 연결은 실패로 간주
  - HSUpSetEnv.exe 를 이용하여 HSUpdate.env 파일에 한 개 이상의 서버를 설정한 경우, 각 서버에 대해 dwTimeOutPerConnection 값을 동일하게 적용
  - '0'으로 설정 시 서버에 연결할 때 대기 시간을 체크하지 않음

## Return Value

### HACKSHIELD\_ERROR\_SUCESS (Value = 0x00000000)

- 설명: 업데이트 성공
- 원인: 정상적인 경우
- 확인사항: N/A

### HSERROR\_ENVFILE\_NOTREAD (Value = 0x30000010)

- 설명: HSUpdate.env 읽기 실패
- 원인: env 환경 파일이 없거나 맞지 않는 버전의 HSUpSetEnv.exe 툴을 사용한 경우
- 확인사항
  - 해당 경로에 env 파일 존재 여부 확인
  - HSUpSetEnv.exe 툴이 sdk 에 존재하는 최신 버전인지 확인

### HSERROR\_ENVFILE\_NOTWRITE (Value = 0x30000020)

- 설명: HSUpdate.env 쓰기 실패

- 원인: HSUpdate.env 파일 생성 시 읽기 속성이거나 접근 권한이 없는 경우
- 확인사항
  - env 환경 파일에 접근 권한이 있는지 확인
  - 환경 파일을 삭제하고 새로 생성했을 때도 동일한 문제가 발생하는지 확인

#### **HSERROR\_NETWORK\_CONNECT\_FAIL (Value = 0x30000030)**

- 설명: 해당 서버 연결 실패
- 원인: 업데이트 서버(ftp/http)에 접속할 수 없는 경우
- 확인사항
  - 업데이트 서버가 정상적으로 접근되는지 ftp 툴이나 Internet Explorer로 확인
  - Env 환경 파일에 업데이트 주소 및 ID 와 비밀번호가 올바르게 입력됐는지 확인
  - 특정 웜의 경우 보안프로그램 사이트 접속을 제한하기 때문에 백신프로그램으로 검사 및 치료
  - 업데이트 서버 최초 구축 시 구축된 서버에서 확장자가 \*.ui 를 인식 못하는 경우에 업데이트 서버에서 해당 파일을 다운로드하지 못하는 경우가 있으므로 업데이트 서버의 파일 구성 요소에 \*.ui 확장자가 인식 가능한지 확인

#### **HSERROR\_LIB\_NOTEDIT\_REG (Value = 0x30000050)**

- 설명: 네트워크 결과 입력 중 에러 발생
- 원인: N/A
- 확인사항: N/A

#### **HSERROR\_NOTFINDFILE (Value = 0x30000060)**

- 설명: HackShield 업데이트 프로그램 관련 파일 찾기 실패
- 원인: N/A
- 확인사항: N/A

#### **HSERROR\_PROTECT\_LISTLOAD\_FAIL (Value = 0x30000070)**

- 설명: HSUpdate.pt 인증 파일 찾기 실패
- 원인: N/A
- 확인사항: N/A

#### **HSERROR\_HSUPDATE\_TIMEOUT (Value = 0x30000090)**

- 설명: HackShield 업데이트 프로그램 업데이트 전송 시간 초과
- 원인: \_AhnHS\_HSUpdate() 호출 시 설정한 타임 아웃 시간 안에 업데이트되지 않았거나 네트워크 상황이 좋지 않은 경우
- 확인사항: N/A

#### **HSERROR\_HSUPDATE\_MODIFICATION (Value = 0x30000091)**

- 설명: HSUpdate.exe 파일이 변경
- 원인: HSUpdate.exe 파일이 HackShield SDK에서 배포하는 파일이 아닌 경우
- 확인사항: N/A

#### **HSError\_HSError\_AUTOUPDATE\_FAIL (Value = 0x300000E0)**

- 설명: HackShield 자동 업데이트 프로세스 중 문제 발생
- 원인: 업데이트 내부 에러일 수 있기 때문에 안랩 HackShield 기술 지원에 문의
- 확인사항: N/A

#### **Remarks**

업데이트 함수는 반드시 초기화 함수 `_AhnHS_Initialize()` 호출 전에 호출해야 합니다. 다른 HackShield 함수와 별도로 게임 프로세스를 실행시키는 런처와 같은 실행 파일에서 호출해서 사용할 수 있습니다.

# **4장**

## **서버 연동 크랙 방지 기능**

- 4.1 서버 연동 크랙 방지 기능 /98**
- 4.2 애플리케이션 프로그래밍 /103**
- 4.3 애플리케이션 프로그래밍 인터페이스 /106**

## 4.1 서버 연동 크랙 방지 기능

HackShield의 서버 연동 크랙 방지 기능을 소개하고 이 기능을 구현하기 위한 방법을 설명합니다.

### 4.1.1 기능 소개

서버 연동(AntiCpX) 크랙 방지 기능은 다음과 같습니다.

- 게임 클라이언트 파일 무결성 검증
- 패킷 무결성
- HackShield 엔진 무결성 확인
- 메모리 무결성 기능
- HackShield 정상 동작 확인

#### 게임 클라이언트 파일 무결성 검증

서버에서 AntiCpXSvr.dll(libanticpxsvr.so, libanticpxsvr\_st.a)과 AntiCpXSvr.h를 적용하고, 클라이언트에서는 HackShield 모듈인 HShield.lib와 HShield.h를 이용하여 서버와 클라이언트 간 통신을 통해 게임 실행 파일의 조작 여부를 실시간으로 확인합니다.

#### ⚠ 주의

게임 클라이언트가 바이러스에 감염되는 경우에도 서버 연동에 의하여 감지될 수 있습니다. 해당 콜백에 대한 종료 처리는 고객의 정책에 맞게 하십시오.

#### 패킷 무결성 검증

패킷을 캡처하고 특정 상황에 맞춰 패킷을 생성하여 정상적으로 동작하게 하는 해킹을 막기 위해 패킷의 무결성을 보장하는 모듈이 내부적으로 동작합니다. 네트워크에서 패킷을 캡처하고 생성하는 것과 같은 방법을 원천적으로 차단할 수 있습니다. 단, 이 기능은 서버 연동 크랙 방지와 관련된 메시지들에 한하여 보호합니다.

#### HackShield 동작 상태 확인

서버에서 AntiCpXSvr.dll (libanticpxsvr.so, libanticpxsvr\_st.a)과 AntiCpXSvr.h를 적용하고, 클라이언트에서는 HackShield 모듈인 HShield.lib와 HShield.h를 이용하여 서버와 클라이언트 간 통신을 통해 간단한 방법으로 HackShield의 동작 상태를 확인할 수 있습니다. 따라서 서버는 HackShield가 동작한 다음에 쿼리 메시지를 전송해야 합니다.

## 메모리 무결성 검증(서버 연동 강화 기능)

서버에서 AntiCpXSvr.dll (libanticpxsvr.so, libanticpxsvr\_st.a)과 AntiCpXSvr.h를 적용하고, 클라이언트에서는 HackShield 모듈인 HShield.lib와 HShield.h를 이용하여 서버와 클라이언트 간 통신을 통해 실시간으로 게임 프로세스의 메모리 조작 여부를 검증할 수 있습니다. 서버 연동에서는 코드 영역 전체를 보호합니다.

## HackShield 엔진 무결성 검증

서버에서 AntiCpXSvr.dll (libanticpxsvr.so, libanticpxsvr\_st.a)과 AntiCpXSvr.h를 적용하고, 클라이언트에서는 HackShield 모듈인 HShield.lib와 HShield.h를 이용하여 서버와 클라이언트 간 통신을 통해 허리스틱 엔진 파일(3n.mhe)의 무결성을 검증할 수 있습니다.

### 4.1.2 기능 특징

서버 연동 크랙 방지 기능의 특징은 다음과 같습니다.

#### 인터페이스 함수(API) 제공

서버 연동(AntiCpX)에서 제공하는 기능을 편리하게 사용하고 그 실행 결과 값을 받아 볼 수 있는 인터페이스 DLL과 라이브러리를 제공합니다. 제공되는 인터페이스 DLL과 라이브러리를 사용하여 게임 개발자는 각 고객의 고유 정책에 따라 게임 파일의 무결성 및 HackShield의 정상 동작 유무를 확인할 수 있습니다.

#### HSB 데이터 파일정보 생성 프로그램 제공

서버 연동(AntiCpX)에서 제공하는 HSBGen.exe는 클라이언트와 서버가 통신하여 클라이언트 프로그램의 파일/메모리 무결성 여부를 판단하는 기준이 되는 각종 파일 정보와 메모리 정보를 생성하고 저장합니다. 고객의 특성에 맞게 서버에서 파일/메모리 정보를 저장할 수 있습니다. 이 데이터를 이용하여 서버 연동을 통한 파일/메모리 크랙 방지 기능이 동작하게 됩니다.

---

#### 참고

자세한 내용은 [9.1 HSBGen 툴](#)을 참고하십시오.

---

### 4.1.3 시스템 구조

서버 연동(AntiCpX)은 AntiCpXSvr.dll(libanticpxsvr.so, libanticpxsvr\_st.a), HShield.lib, EHService.dll을 제공하여 서버와 클라이언트에 DLL과 라이브러리 형태로 적용됩니다. 서버 연동(AntiCpX)의 전체 구조 및 동작 원리는 다음과 같습니다.



그림 16 AntiCpX의 동작 원리

위 그림에서 보면 게임 서버가 최초 실행될 때 `_AhnHS_CreateServerObject()`를 통해 AHNHS\_SERVER\_HANDLE을 생성합니다. 이후 각 클라이언트가 접속 될 때마다 `_AhnHS_CreateClientObject()`와 서버 핸들을 파라미터로 취하여 AHNHS\_CLIENT\_HANDLE을 생성합니다. 클라이언트의 위변조 여부를 감시하기 위해 `_AhnHS_MakeRequest()`를 사용하여 요청 메시지를 주기적으로 생성하여 전송합니다.

클라이언트는 서버의 요청 메시지에 적절한 응답 메시지를 생성합니다. 응답 메시지는 `_AhnHS_MakeResponse()`를 통해 생성합니다. 클라이언트의 응답 메시지가 유효한지는 `_AhnHS_VerifyResponseEx()`를 통해 검사합니다.

### ⚠ 주의

`_AhnHS_MakeResponse()`에 의해 생성된 응답메시지가 서버로 전달되어 `_AhnHS_VerifyResponseEx()`가 호출되기 전까지 `_AhnHS_MakeRequest()`가 호출되지 않도록 구성합니다.

위 그림의 주요 항목에 대한 설명은 다음과 같습니다.

### **AntiCpXSvr.dll(인터페이스 dll)**

Windows용 동적 라이브러리 파일입니다. 서버에서 사용되며 서버에서 요청 메시지를 생성하고 클라이언트에서 받은 메시지를 이용하여 클라이언트 파일 및 메모리가 정상 동작하는지 확인할 수 있게 하는 API를 제공합니다.

### **AntiCpXSvr.lib**

Windows용 정적 라이브러리 파일입니다. 서버에서 사용되며 서버에서 요청 메시지를 생성하고 클라이언트에서 받은 메시지를 이용하여 클라이언트 파일 및 메모리가 정상 동작하는지 확인할 수 있게 하는 API를 제공합니다.

### **libanticpxsvr.so**

Linux나 Unix용(Solaris용) 동적 라이브러리 파일입니다. 서버에서 사용되며 서버에서 요청 메시지를 생성하고 클라이언트에서 받은 메시지를 이용하여 클라이언트 파일 및 메모리가 정상 동작하는지 확인할 수 있게 하는 API를 제공합니다.

### **libanticpxsvr\_st.a**

Linux용 및 Solaris용 정적 라이브러리 파일입니다. 서버에서 사용되며 서버에서 요청 메시지를 생성하고 클라이언트에서 받은 메시지를 이용하여 클라이언트 파일 및 메모리가 정상 동작하는지 확인할 수 있게 하는 API를 제공합니다.

### **HShield.lib(HackShield 라이브러리)**

클라이언트에서 사용되며 서버에서 전송한 요청 메시지를 받아 요구한 내용을 처리하고 결과로 얻은 각종 정보를 암호화하여 서버에 전송할 데이터를 만드는 API를 제공합니다.

### **HSBGen.exe(파일 정보 생성 프로그램)**

서버에서 사용할 파일 정보 및 메모리 정보 파일을 생성하여 게임 파일의 무결성 여부와 실행되는 게임의 메모리 무결성 여부를 판단할 수 있게 합니다. 높은 보안을 유지하려면 파일 정보 데이터를 생성할 때마다 매번 다른 값을 저장하여, 게임 서버가 여러 개이면 서버마다 다르게 적용합니다.

### **AntiCpx.hsb(클라이언트 CRC 파일)**

HSBGen.exe 툴에서 생성되는 파일입니다. 클라이언트 프로그램에 대해 무결성을 보장할 수 있게 합니다.

## HSPub.key(서버 연동 인증 키 파일)

접속하는 클라이언트의 HackShield 인증 용도로 사용합니다. AntiCpx.hsb와 동일한 경로에 있어야 합니다.

## 3n.mhe(휴리스틱 엔진 파일)

클라이언트에서 사용하는 엔진 파일입니다. 서버 연동 사용 시 이전 버전의 3n.mhe 엔진을 사용하는 클라이언트의 접속을 끊을 수 있게 서버의 AntiCpX.hsb와 동일한 경로에 놓아야 합니다.

## hshield.dat(HackShield 버전 파일)

클라이언트에서 사용하는 데이터 파일입니다. 서버 연동 사용 시 이전 버전의 hshield.dat 파일을 사용하는 클라이언트의 접속을 끊을 수 있도록 서버의 AntiCpX.hsb와 동일한 경로에 놓아야 합니다.

### 참고

서버 hsb 파일 경로에 3n.mhe이나 hshield.dat 파일을 놓아두면 이전 버전의 HackShield로 접속하는 클라이언트 접속을 끊을 수 있습니다. (3n.mhe/hshield.dat 파일을 올려 놓을 때는 서버 재기동이 필요 없음)

다음 표와 같이 접속 끊음으로 표시되는 경우에 한해서 \_AhnHS\_VerifyResponseEx()에서 ANTICPX\_RECOMMAND\_CLOSE\_SESSION

(세부 에러: ERROR\_ANTICPXSvr\_INVALID\_ENGINE\_VERSION,

ERROR\_ANTICPXSvr\_INVALID\_HACKSHIELD\_VERSION) 에러가 리턴됩니다.

해당 에러가 리턴될 경우 클라이언트 접속을 끊으시면 됩니다. 표 13을 참고하십시오.

표 13 서버 연동 버전 관리

| 파일 이름       | 버전 비교      | 접속 상태 |
|-------------|------------|-------|
| 3n.mhe      | 클라이언트 < 서버 | 접속 끊음 |
|             | 클라이언트 ≥ 서버 | 접속 유지 |
| hshield.dat | 클라이언트 < 서버 | 접속 끊음 |
|             | 클라이언트 ≥ 서버 | 접속 유지 |

### 참고

서버 관련 API 내에서 기능 실행 중 예외(Exception) 발생 시 덤프 파일을 생성할 수 있습니다. 레지스트리 값 설정으로 해당 기능을 실행할 수 있으며 기본 값으로는 덤프 파일을 남기지 않도록 설정되어 있습니다. 단, 이는 Windows에서만 동작됩니다.

HKLM\Software\AhnLab\HShield\Dump 키 값 내 관련된 설정 값은 다음과 같습니다.

-DumpOnOff: DWORD 값이며 해당 값이 '0x1'이어야 덤프 파일 남기는 것이 가능. 해당 값이 '0x0'이면 덤프 파일을 남기지 않음

-DumpType: DWORD 값이며 MINIDUMP\_TYPE 구조체에 정의된 값을 참고하여 설정. 자세한 내용은 MSDN Library 참고. 예외 발생에 대한 정확한 분석을 위해서는 'MiniDumpWithFullMemory' 값인 '0x2'값으로 설정할 것을 권장. 이후에 해당 덤프 파일의 크기가 커져 디스크 용량을 많이 차지

---

한다면 'MiniDUMPNormal' 값인 '0x1'로 변경

-DumpPath: DumpPath 값은 문자열 값이며 덤프 파일이 생성될 디렉터리 경로를 지정. 해당 경로는 접근 가능한 경로여야 함. 만약, DumpPath 값이 생성되어 있지 않거나 해당 값 내의 경로가 접근 가능하지 않은 경로이거나 잘못된 경로인 경우에는 현재 게임 서버가 실행된 경로 내에 덤프 파일이 남음

HKLM\Software\AhnLab\HShield\Dump는 예외가 발생하면 생성됩니다. DumpOnOff, DumpType, DumpPath는 직접 생성하고 입력해야 하는 값들입니다.

---

## 4.2 애플리케이션 프로그래밍

---

HackShield의 서버 연동 크랙 방지 기능을 구현하는 방법을 설명합니다.

---

### 참고

이 문서에서 사용하고 있는 샘플 코드는 Microsoft Visual C++ 6.0을 기준으로 한 C/C++ 언어로 작성했습니다. 프로그래밍에 사용되는 언어는 각 프로그램의 특성 및 시스템 환경에 따라서 변경 할 수 있습니다.

---

### 4.2.1 프로그래밍 준비

서버 연동(AntiCpX)을 사용하여 프로그래밍을 시작하기 전에 확인해야 하는 서버 연동 관련 구성 파일 목록과 AntiCpXSvr 관련 파일 목록은 다음과 같습니다.

#### 서버 연동 관련 구성 파일

서버 연동 관련 구성 파일은 다음과 같습니다.

표 14 서버 연동 관련 구성 파일

| 파일 이름                                            | 설명                                 |
|--------------------------------------------------|------------------------------------|
| \Bin\Win\[Platform]\AntiCrack\AntiCpXSvr.dll     | 서버 연동 dll 파일.                      |
| \Bin\Win\x86\AntiCrack\HSBGen.exe                | *.hsb 생성 툴                         |
| \Bin\Win\x86\AntiCrack\HSPub.key                 | 서버 연동 인증 키 파일                      |
| \Bin\Win\x86\HShield\3n.mhe                      | 휴리스틱 엔진 버전 관리 파일                   |
| \Bin\Win\x86\HShield\hshield.dat                 | HackShield 버전 관리 파일                |
| \Include\AntiCpXSvr.h                            | 서버 연동 헤더 파일                        |
| \Lib\Win\x86\AntiCpXSvr.lib                      | 서버 연동 라이브러리 파일<br>(Windows용 32Bit) |
| \Lib\Win\x64\AntiCpXSvr.lib                      | 서버 연동 라이브러리 파일<br>(Windows용 64Bit) |
| \Bin\Linux\[Platform]\AntiCrack\libanticpxsvr.so | 서버 연동 동적 라이브러리 파일<br>(Linux용)      |
| \Bin\Solaris\x86\AntiCrack\libanticpxsvr.so      | 서버 연동 동적 라이브러리 파일<br>(Solaris용)    |

|                                               |                                 |
|-----------------------------------------------|---------------------------------|
| \Lib\Linux\[Platform]\libanticpxsvr_st.a      | 서버 연동 정적 라이브러리 파일<br>(Linux용)   |
| \Lib\Solaris\x86\AntiCrack\libanticpxsvr_st.a | 서버 연동 정적 라이브러리 파일<br>(Solaris용) |

※ Solaris용 파일은 32bit x86 플랫폼만 지원

### AntiCpXSvr 관련 파일

AntiCpXSvr 관련 파일은 다음과 같습니다.

**표 15 AntiCpXSvr 관련 파일**

| 파일 이름              | 설치 폴더                                     | 설명                                       |
|--------------------|-------------------------------------------|------------------------------------------|
| AntiCpXSvr.h       | [프로그램 소스 폴더]                              | 서버에서 사용할 헤더 파일                           |
| AntiCpXSvr.lib     | [프로그램 소스 폴더]                              | AntiCpXSvr.dll의 임포트 라이브러리<br>(windows용)  |
| AntiCpXSvr.dll     | [프로그램 실행 폴더]]                             | 서버에서 사용할 DLL 파일                          |
| Libanticpxsvr.so   | [프로그램 실행 폴더]                              | Linux/Solaris용 서버 모듈<br>(동적 라이브러리 파일)    |
| Libanticpxsvr_st.a | [프로그램 실행 폴더]                              | Linux/Solaris용 서버 모듈<br>(정적 라이브러리 파일)    |
| AntiCpx.hsb        | [AhnHS_CreateServerObject API 파라미터 경로 폴더] | 클라이언트 무결성 검증 파일                          |
| HSPub.key          | [HSB파일과 동일한 폴더]                           | 서버 연동 인증 키 파일                            |
| 3n.mhe             | [HSB파일과 동일한 폴더]                           | HackShield 휴리스틱 엔진 버전 관리 파일              |
| hshield.dat        | [HSB파일과 동일한 폴더]                           | HackShield 버전 관리 파일                      |
| HShield.h          | [프로그램 소스 폴더]                              | 클라이언트에서 사용할 헤더 파일<br>HackShield 기능 포함    |
| HShield.lib        | [프로그램 소스 폴더]                              | 클라이언트에서 사용할 라이브러리 파일<br>HackShield 기능 포함 |
| EHSvc.dll          | [프로그램 소스 폴더]                              | 클라이언트에서 사용할 DLL 파일<br>HackShield 기능 포함   |

### 4.2.2 프로그래밍 적용

프로그래밍을 하기 위해 서버와 클라이언트를 준비하고 적용하는 방법과 서버 연동 주기를 설정하는 방법은 다음과 같습니다.

#### 서버 적용 방법

서버 적용 방법은 다음과 같습니다.

1. HSBGen.exe를 이용하여 AntiCpx.hsb를 생성합니다.

---

### 참고

자세한 내용은 [9.1 HSBGen 툴](#)을 참고하십시오.

---

2. HSPub.key 파일을 AntiCpx.hsb 파일이 존재하는 동일한 폴더에 복사합니다.
  3. 3n.mhe 와 hshield.dat 파일을 hsb 파일이 존재하는 폴더에 복사합니다. (필수 과정은 아니지만 HackShield 버전 관리를 위해 권장)
- 

### 참고

서버에 hsb가 존재하는 경로에 클라이언트에서 사용하는 3n.mhe와 hshield.dat 파일을 옮겨두어 3n.mhe와 hshield.dat 버전이 올바르지 않은 클라이언트 접속을 차단할 수 있습니다.

---

4. 게임 서버가 최초로 실행될 때 \_AhnHS\_CreateServerObject()를 통해 AHNHS\_SERVER\_HANDLE 을 생성합니다. 이 핸들은 게임 서버가 종료되기 전까지 유지합니다. (서버 핸들은 순서 5 의 클라이언트 핸들을 만들 때 사용됨)
5. 각 클라이언트가 접속될 때 마다 \_AhnHS\_CreateClientObject()와 서버 핸들을 파라미터로 받아 AHNHS\_CLIENT\_HANDLE 을 생성합니다. 이 핸들은 클라이언트 네트워크 접속이 유지하는 세션 동안 유지합니다. (클라이언트 핸들은 순서 6 의 요청 메시지를 만들 때 사용됨)
6. 클라이언트의 위변조 여부를 감시하기 위해 요청 메시지를 생성하여 전송합니다. 요청 메시지는 \_AhnHS\_MakeRequest()를 사용하여 생성하며 클라이언트 핸들을 파라미터로 사용합니다.
7. 클라이언트는 서버의 요청 메시지에 적절한 응답 메시지를 생성합니다. 응답 메시지는 \_AhnHS\_MakeResponse()를 통해 생성합니다.
8. 클라이언트의 응답 메시지가 유효한지 검사합니다. 응답 메시지의 유효성 검사는 \_AhnHS\_VerifyResponseEx()를 통해 검사합니다.
9. \_AhnHS\_VerifyResponseEx()에서 ANTICPX\_RECOMMAND\_CLOSE\_SESSION 값이 발생했을 경우 인자로 전달 받은 여러 값에 따라 적절하게 에러를 처리한 후 게임 클라이언트의 접속을 중단합니다.

---

### 주의

\_AhnHS\_VerifyResponseEx()와 \_AhnHS\_VerifyResponse() 중에 선택하여 적용하십시오.

---

10. 클라이언트 접속을 끊을 때, 즉 세션이 종료될 때 순서 5 에서 생성한 클라이언트 핸들을 닫습니다.
11. 서버 프로세스가 종료될 때 순서 4 에서 생성한 서버 핸들을 닫습니다. 이 때 클라이언트 핸들은 모두 닫힌 상태여야 합니다.

## 클라이언트 적용 방법

클라이언트 적용 방법은 다음과 같습니다.

1. HShield.lib 파일을 작업할 프로젝트에 포함시킵니다.
2. 제공되는 HShield.h 파일을 작업할 소스 파일에 포함시킵니다.
3. 서버로부터 암호화된 버전 요청 메시지를 받습니다.
4. \_AhnHS\_MakeResponse()를 호출하여 서버로 전달한 암호화된 버전 응답 메시지를 생성합니다.
5. 암호화된 응답 메시지를 서버에 전송합니다.

## 서버 연동 주기

서버에서 요청 메시지를 보내고 클라이언트에서 응답 메시지를 생성한 후 서버에서 받는 것을 하나의 서버 연동 주기로 설정합니다. 주기 사이 간격은 1분 이내로 설정할 것을 권장합니다. 게임의 특성에 따라 설정이 가능하지만 서버 연동의 주기가 길어지면 해킹 감지 시간이 길어질 수 있기 때문입니다.

서버 연동 주기를 그림으로 나타내면 다음과 같습니다.

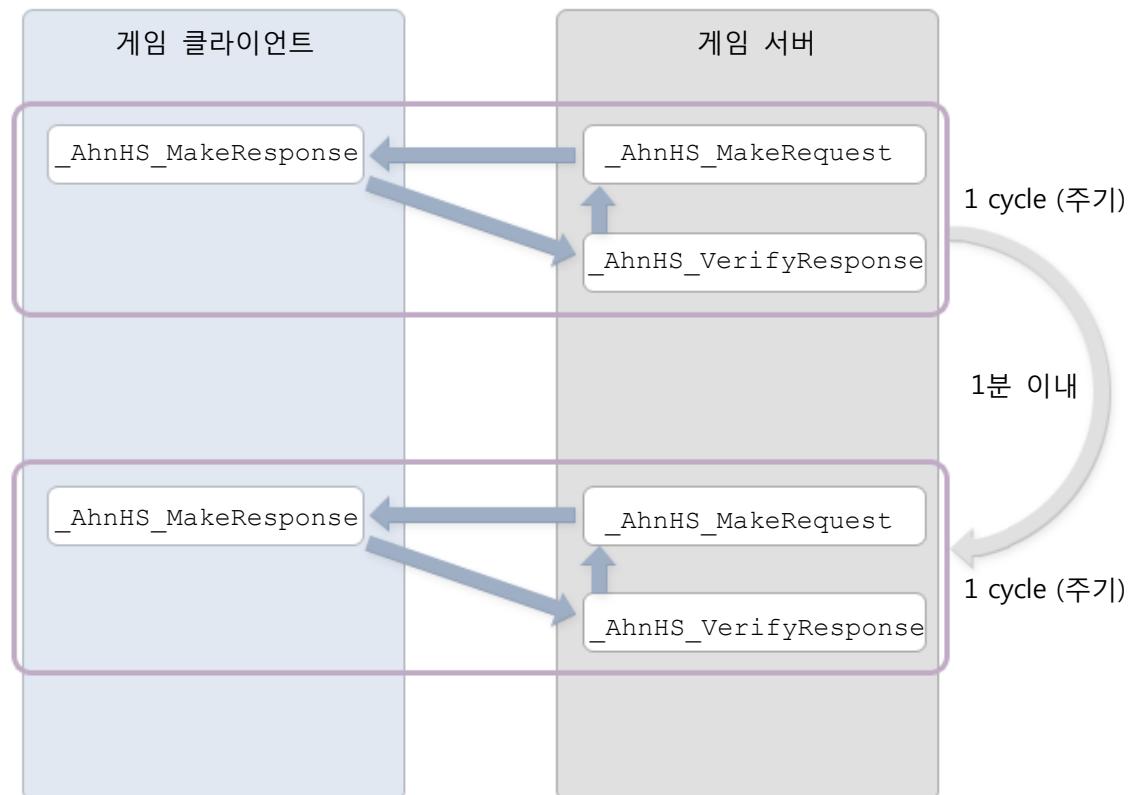


그림 17 서버 연동 주기

## 4.3 애플리케이션 프로그래밍 인터페이스

HackShield의 서버 연동 크랙 방지 기능 구현에 필요한 API를 소개합니다.

### 4.3.1 \_AhnHS\_CreateServerObject()

HSBGen.exe를 통해 생성한 .hsb 파일을 로드하여 서버 핸들(Server Handle)을 생성하는 함수입니다. 보통 하나의 게임을 서비스하는 서버 프로세스에서 서버 핸들은 한 개를 생성하여 게임 서버 프로세스가 종료할 때까지 유지합니다.

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

#### Parameters

| Parameter   | Value        | Description                          |
|-------------|--------------|--------------------------------------|
| pszFilePath | const char * | HackShield Briefcase (.hsb) 파일 전체 경로 |

#### Return Value

##### ANTICPX\_INVALID\_HANDLE\_VALUE

- 설명: 서버 핸들을 올바르게 생성하지 않음
- 원인
  - HackShield Briefcase (.hsb) 파일 경로가 올바르지 않거나 파일이 없는 경우
  - HSPub.key 파일이 해당 경로에 존재하지 않을 경우
  - 시스템 리소스(메모리)가 부족할 경우
  - hsb 파일이 존재하지 않거나 접근 권한이 없는 경우
  - \_AhnHS\_CreateServerObject() 동작 중 예외(Exception)가 발생한 경우
  - 구 버전의 HSBGen.dll (1.0.19.0 이하 버전)을 이용하여 생성한 hsb 파일이 사용된 경우
- 확인사항: 예외가 발생한 경우, Windows 환경에서만 내부적으로 MiniDump 파일을 남기도록 되어 있음. 'CREATESERVEROBJECT\_년\_월\_일 시-분-초.dmp' 형식으로 생성된 덤프 파일을 안랩 HackShield 기술 지원에 전달

#### Example

\_AhnHS\_CreateServerObject() 호출 예제는 다음과 같습니다.

```
// 파일 경로는 '/' 아닌 '\'으로 표현해야 합니다.  
// 파일 이름까지 경로를 표현해줍니다.  
  
strcpy(g_szHsbFilePath, "C:\\GameServer\\HShield\\anticpx.hsb");  
  
hServer = _AhnHS_CreateServerObject (g_szHsbFilePath);
```

```

if ( hServer == ANTICPX_INVALID_HANDLE_VALUE )
{
    // 에러 처리
}

```

그림 18 \_AhnHS\_CreateServerObject() 호출 예제

#### 4.3.2 \_AhnHS\_CloseServerHandle()

서버 핸들(Server Handle)을 닫는 함수입니다.

```

void __stdcall
_AhnHS_CloseServerHandle (
    IN AHNHS_SERVER_HANDLE hServer
);

```

##### Parameters

| Parameter   | Value               | Description                            |
|-------------|---------------------|----------------------------------------|
| pszFilePath | AHNHS_SERVER_HANDLE | _AhnHS_CreateServerObject()를 통해 생성한 핸들 |

##### Return Value

N/A

##### Example

\_AhnHS\_CloseServerHandle() 호출 예제는 다음과 같습니다.

```

_AhnHS_CloseServerHandle ( hServer );

```

그림 19 \_AhnHS\_CloseServerHandle() 호출 예제

#### 4.3.3 \_AhnHS\_CreateClientObject()

서버 핸들을 입력 받아 클라이언트 핸들을 생성하는 함수입니다. 클라이언트 핸들은 클라이언트가 접속할 때마다 생성하며 세션이 유지되는 동안 핸들을 닫지 않고 재사용합니다.

```

AHNHS_CLIENT_HANDLE
_AhnHS_CreateClientObject (
    IN AHNHS_SERVER_HANDLE hServer
);

```

## Parameters

| Parameter | Type                | Description          |
|-----------|---------------------|----------------------|
| hServer   | AHNHS_SERVER_HANDLE | 서버 핸들(Server Handle) |

## Return Value

### ANTICPX\_INVALID\_HANDLE\_VALUE

- 설명: 서버 핸들을 올바르게 생성되지 않음
- 원인: 전달된 hServer 인자가 유효하지 않은 경우, \_AhnHS\_CreateServerObject() 동작 중 예외(Exception)가 발생한 경우
- 확인사항: hServer 유효성이 원인인 경우에는 \_AhnHS\_CreateServerObject()를 통해 hServer 서버 핸들이 정상적으로 생성 되었는지 확인. 예외가 발생한 경우에는 Windows 환경에서만 내부적으로 MiniDump 파일을 남기도록 되어 있음.  
'CREATESERVEROBJECT\_년\_월\_일 시-분-초.dmp' 형식으로 생성된 덤프 파일을 안랩 HackShield 기술 지원에 전달

## Example

\_AhnHS\_CreateClientObject() 호출 예제는 다음과 같습니다.

```
hClient = _AhnHS_CreateClientObject ( hServer );  
  
if ( hClient == ANTICPX_INVALID_HANDLE_VALUE )  
{  
    // 에러 처리  
}
```

그림 20 \_AhnHS\_CreateClientObject() 호출 예제

### 4.3.4 \_AhnHS\_CloseClientHandle()

클라이언트 세션 종료 시 생성했던 클라이언트 핸들도 종료하기 위해 클라이언트 핸들 생성에 사용했던 메모리나 시스템 자원을 해제하는 함수입니다.

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

## Parameters

| Parameter | Type                | Description             |
|-----------|---------------------|-------------------------|
| hClient   | AHNHS_CLIENT_HANDLE | 클라이언트 핸들(Client Handle) |

## Return Value

N/A

## Example

\_AhnHS\_CloseClientHandle() 호출 예제는 다음과 같습니다.

```
_AhnHS_CloseClientHandle ( hClient );
```

그림 21 \_AhnHS\_CloseClientHandle()

### 4.3.5 \_AhnHS\_MakeRequest()

현재 세션에 맞는 클라이언트 핸들을 입력하여 요청 메시지를 생성하는 함수입니다. 요청 메시지는 \_AHNHS\_TRANS\_BUFFER 구조체 형태로 출력됩니다.

\_AHNHS\_TRANS\_BUFFER는 다음과 같습니다.

```
typedef struct _AHNHS_TRANS_BUFFER
{
    unsigned short nLength;
    // 송수신 패킷의 최대 버퍼 크기
    unsigned char byBuffer[ANTICPX_TRANS_BUFFER_MAX];
} AHNHS_TRANS_BUFFER, *PAHNHS_TRANS_BUFFER;
```

AHNHS\_TRANS\_BUFFER의 멤버 변수에 대한 설명은 다음과 같습니다.

- nLength: 요청 메시지 생성에 사용된 버퍼 길이
- byBuffer: 요청 메시지 생성에 사용될 수 있는 최대 바이트 버퍼

---

#### ⚠ 주의

byBuffer는 요청 메시지 생성에 사용될 수 있는 최대 버퍼 크기이므로 네트워크로 전송할 때 nLength 만큼만 전송해야 합니다.

---

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

## Parameters

| Parameter | Type                | Description |
|-----------|---------------------|-------------|
| hClient   | AHNHS_CLIENT_HANDLE | 클라이언트 핸들    |

|                |                     |              |
|----------------|---------------------|--------------|
| pRequestBuffer | PAHNHS_TRANS_BUFFER | 보낼 데이터 버퍼/길이 |
|----------------|---------------------|--------------|

## Return Value

### **ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

### **ERROR\_ANTICPXSVR\_INVALID\_PARAMETER (Value = 0xE9040003)**

- 설명: 잘못된 입력 값
- 원인: hClient, pRequestBuffer 값이 NULL 인 경우
- 확인사항: hClient, pRequestBuffer 값의 NULL 여부 확인

### **ERROR\_ANTICPXSVR\_BAD\_FORMAT (Value = 0xE9040004)**

- 설명: HSB 파일 읽기 에러
- 원인: HSB 파일이 정상적으로 생성이 되지 않은 경우
- 확인사항: HSBGen.exe 를 최신 버전으로 사용했는지 확인해 보고 HSB 파일 다시 생성

### **ERROR\_ANTICPXSVR\_NOT\_YET\_RECEIVED\_RESPONSE (Value = 0xE9040005)**

- 설명: 이전 요청 메시지에 대한 응답을 받지 않음
- 원인: \_AhnHS\_MakeRequest()를 호출하면 클라이언트로부터 Ack 메시지를 받아서 \_AhnHS\_VerifyResponse()를 거쳐서 다시 호출되는 동기화 과정이 정상적으로 이루어지지 않은 경우
- 확인사항: 클라이언트 세션 관리 상의 동기화 문제가 발생하지 않았는지 확인

### **ERROR\_ANTICPXSVR\_NOT\_ENOUGH\_MEMORY (Value = 0xE9040007)**

- 설명: 메모리 부족
- 원인: 서버 메모리가 부족한 경우
- 확인사항: 서버에서 메모리 leak 현상이 발생하는지 확인

### **ERROR\_ANTICPXSVR\_BAD\_MESSAGE (Value = 0xE9040008)**

- 설명: 버퍼 암호화 실패 또는 버퍼의 크기가 정상적이지 않은 경우.
- 원인: 요청 보낼 버퍼를 암호화하는데 실패하거나, 버퍼의 크기가 작은 경우.
- 확인사항: 구조적인 문제일 수 있기 때문에 안랩 HackShield 기술 지원에 문의

### **ERROR\_ANTICPXSVR\_MAKEREQ\_EXCEPTION (Value = 0xE9040014)**

- 설명: 예외(Exception) 발생(Windows 용 모듈에서만 발생)

- 원인: 여러 가지 원인
- 확인사항: 예외가 발생한 경우, Windows 환경에서만 내부적으로 MiniDump 파일을 남기도록 되어 있음. 'MAKEREQUEST\_년\_월\_일 시-분-초.dmp' 형식으로 생성된 덤프 파일을 안랩 HackShield 기술 지원에 전달

## Example

`_AhnHS_MakeRequest()` 호출 예제는 다음과 같습니다.

```
AHNHS_TRANS_BUFFER stReqTransBuf;

ulRet = _AhnHS_MakeRequest ( hClient, &stReqTransBuf );

if ( ulRet != ERROR_SUCCESS )
    return ulRet;

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

...
```

그림 22 `_AhnHS_MakeRequest()` 호출 예제

### ⚠ 주의

게임 서버에 게임 유저가 접속한 후 바로 `_AhnHS_MakeRequest()`를 호출해서 해킹 감지도 바로 이루어지도록 하십시오.

### 4.3.6 `_AhnHS_VerifyResponseEx()`

`_AhnHS_MakeRequest()`를 통한 요청 메시지의 클라이언트 응답이 올바른지 검사하는 함수입니다. 내부적으로 `_AhnHS_VerifyResponse()`를 호출하도록 되어 있습니다.

```
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 | 클라이언트 핸들    |

|                 |               |                             |
|-----------------|---------------|-----------------------------|
| pbyResponse     | char *        | 클라이언트에게 받은 데이터 버퍼           |
| nResponseLength | unsigned long | 클라이언트에게 받은 데이터 길이           |
| pnErrorCode     | Unsigned long | _AhnHS_VerifyResponse() 반환값 |

## Return Value

### ANTICPX\_RECOMMAND\_CLOSE\_SESSION (Value = 101)

- 설명: 클라이언트에서 해킹 행위가 감지되었으므로 게임 서버에서 해당 게임 클라이언트와의 접속 종료. 해킹 행위는 주로 \_AhnHS\_VerifyResponse()에서 설명한 여러 코드 중에 클라이언트와의 접속을 종료하도록 권장한 에러가 리턴되는 경우. 또는 내부적으로 \_AhnHS\_VerifyResponse()를 호출하여 해당 함수의 반환값이 다음 값들 중 하나인 경우 ANTICPX\_RECOMMAND\_CLOSE\_SESSION 을 리턴
  - 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\_ANORMAL\_HACKSHIELD\_STATUS
  - ERROR\_ANTICPXSvr\_DETECT\_CALLBACK\_IS\_NOTIFIED
  - -ERROR\_ANTICPXSvr\_OLD\_HACKSHIELD\_VERSION
  - -ERROR\_ANTICPXSvr\_ANORMAL\_HACKSHIELD\_XSTATUS
- 원인: 클라이언트에서 해킹 행위가 감지된 경우
- 확인사항: N/A

### ANTICPX\_RECOMMAND\_KEEP\_SESSION (Value = 102)

- 설명: 정상적인 클라이언트 응답. 게임 서버에서 해당 게임 클라이언트와의 접속을 계속 진행
- 원인: N/A
- 확인사항: N/A

### ERROR\_ANTICPXSvr\_VERIFYEX\_EXCEPTION (Value = 0xE904001A)

- 설명: 예외(Exception) 발생. 단, Windows 용 모듈에서만 발생
- 원인: 여러 가지 원인

- 확인사항: 예외가 발생한 경우, Windows 환경에서만 내부적으로 MiniDump 파일을 남기도록 되어 있음. 'VERIFYRESPONSEEX\_년\_월\_일 시-분-초.dmp' 형식으로 생성된 덤프 파일을 안랩 HackShield 기술 지원에 전달

## Example

`_AhnHS_VerifyResponseEx()` 호출 예제는 다음과 같습니다.

```

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 -> _AhnHS_CloseClientHandle를
    // 호출하여 hClient 핸들을 닫아줌

}

```

그림 23 `_AhnHS_VerifyResponseEx()` 호출 예제

### 4.3.7 `_AhnHS_VerifyResponseEx_WithInfo()`

`_AhnHS_MakeRequest()`를 통한 요청 메시지의 클라이언트 응답이 올바른지 검사하는 함수입니다. 내부적으로 `_AhnHS_VerifyResponse()`를 호출하도록 되어 있습니다.

```

unsigned long __stdcall
_AhnHS_VerifyResponseEx_WithInfo (
    IN AHNHS_CLIENT_HANDLE hClient,
    IN unsigned char *pbyResponse,
    IN unsigned long nResponseLength,
    OUT unsigned long *pnErrorCode,
    OUT unsigned long *pnSpecificError
);

```

## Parameters

| Parameter       | Type                | Description        |
|-----------------|---------------------|--------------------|
| hClient         | AHNHS_CLIENT_HANDLE | 클라이언트 핸들           |
| pbyResponse     | char *              | 클라이언트로부터 받은 데이터 버퍼 |
| nResponseLength | unsigned long       | 클라이언트로부터 받은 데이터 길이 |

|                 |                |                              |
|-----------------|----------------|------------------------------|
| pnErrorCode     | unsigned long* | _AhnHS_VerifyResponse() 반환값  |
| pnSpecificError | unsigned long* | pnErrorCode 에러 값을 보충하는 에러 코드 |

## Return Value

### ANTICPX\_RECOMMAND\_CLOSE\_SESSION (Value = 101)

- 설명: 클라이언트에서 해킹 행위가 감지되었으므로 게임 서버에서 해당 게임 클라이언트와의 접속을 종료해야 함(해킹 행위는 주로 \_AhnHS\_VerifyResponse()에서 설명한 에러 코드 중에 클라이언트와의 접속을 종료하도록 권장한 에러가 리턴되는 경우)
 

내부적으로 \_AhnHS\_VerifyResponse()를 호출하여 해당 함수의 반환값이 다음 값들 중 하나일 경우 ANTICPX\_RECOMMAND\_CLOSE\_SESSION 을 리턴

  - 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
  - ERROR\_ANTICPXSvr\_OLD\_HACKSHIELD\_VERSION
  - ERROR\_ANTICPXSvr\_ABNORMAL\_HACKSHIELD\_XSTATUS
- 원인: 클라이언트에서 해킹 행위가 감지된 경우
- 확인사항: N/A

### ANTICPX\_RECOMMAND\_KEEP\_SESSION (Value = 102)

- 설명: 정상적인 클라이언트 응답. 게임 서버에서 해당 게임 클라이언트와의 접속을 계속 진행
- 원인: N/A
- 확인사항: N/A

### ERROR\_ANTICPXSvr\_VERIFYEX\_EXCEPTION (Value = 0xE904001A)

- 설명: 예외(Exception) 발생. 단, Windows용 모듈에서만 발생
- 원인: 여러 가지 원인

- 확인사항: 예외가 발생한 경우, Windows 환경에서만 내부적으로 MiniDump 파일을 남기도록 되어 있음. 'VERIFYRESPONSEEX\_년\_월\_일 시-분-초.dmp' 형식으로 생성된 덤프 파일을 안랩 HackShield 기술 지원에 전달

## Remarks

Specific Error Code는 Error Code를 보충 설명하는 에러 코드로, 에러 상황을 좀 더 명확하게 파악할 수 있습니다. 에러 코드에 대한 상세 에러 코드 설명은 다음과 같습니다. 단, Specific Error Code는 현재 ERROR\_ANTICPXSvr\_DETECT\_CALLBACK\_IS\_NOTIFIED 에러 코드에 대해서만 의미가 있습니다.

**표 16 Specific Error Code**

| Error Code                                   | Specific Error Code                                                                                                                                                                         |
|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ERROR_ANTICPXSvr_DETECT_CALLBACK_IS_NOTIFIED | AntiCpXSvr.h에 정의된 Client-Side CallBack Code<br>(AHNHS_ACTAPC_DETECT_SPEEDHACK,<br>AHNHS_ENGINE_DETECT_GAME_HACK,<br>AHNHS_ACTAPC_DETECT_MULTI_LOADING,<br>AHNHS_ACTAPC_DETECT_AUTOMOUSE...) |

## Example

\_AhnHS\_VerifyResponseEx\_WithInfo() 호출 예제는 다음과 같습니다.

```
// unsigned long은 linux 64bit에서는 64bit 이므로, 32bit 변수로
// 사용하지 않도록 주의해야 합니다.

unsigned long ulError = 0;
unsigned long ulSpecificError = 0;
unsigned long ulRet = ERROR_SUCCESS;
ulRet = _AhnHS_VerifyResponseEx_WithInfo ( hClient,
   stResponseBuf.byBuffer,
   stResponseBuf.nLength,
   &ulError,
   &ulSpecificError);

if ( ulRet == ANTICPX_RECOMMAND_CLOSE_SESSION )
{
    BOOL bKick = false;
    if ( ulError == ERROR_ANTICPXSvr_DETECT_CALLBACK_IS_NOTIFIED )
    {
        switch (ulSpecificError)
        {
            case : AHNHS_ACTAPC_DETECT_MULTI_LOADING:
                // 클라이언트에서 멀티 로딩이 이루어지는 경우 접속을 끊음
                bKick = true;
                break;
        }
    }
}
```

```

    ...
    default:
        break;

    }

    if ( true = bKick )
    {
        Log ("[AVREx] Disconnect the Client: 0x%x (0x%x)", ulError,
ulSpecificError);
        // Client Out -> _AhnHS_CloseClientHandle를
        // 호출하여 hClient 핸들을 닫아줌
    }
}

```

**그림 24 \_AhnHS\_VerifyResponseEx\_WithInfo() 호출 예제**

#### 4.3.8 \_AhnHS\_VerifyResponse()

\_AhnHS\_MakeRequest()를 통한 요청 메시지의 클라이언트 응답이 올바른지 검사하는 함수입니다.

```

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 | 클라이언트 핸들          |
| pbyResponse     | char *              | 클라이언트에게 받은 데이터 버퍼 |
| nResponseLength | unsigned long       | 클라이언트에게 받은 데이터 길이 |

#### Return Value

**ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

#### **ERROR\_ANTICPXSVR\_INVALID\_PARAMETER (Value = 0xE9040003)**

- 설명: 잘못된 입력 값
- 원인: hClient, pbyResponse 값이 NULL 인 경우
- 확인사항: hClient, pbyResponse 값의 NULL 여부 확인

#### **ERROR\_ANTICPXSVR\_BAD\_FORMAT (Value = 0xE9040004)**

- 설명: 잘못된 포맷
- 원인: 복호화 과정에서 실패가 일어난 경우
- 확인사항: 서버에서 보낸 버퍼가 정상적으로 클라이언트의 \_AhnHS\_VerifyResponse()에 전달되었는지 확인

#### **ERROR\_ANTICPXSVR\_NO\_WAITING (Value = 0xE9040006)**

- 설명: 이전 요청 메시지에 대한 응답을 받지 않음
- 원인: \_AhnHS\_MakeRequest()를 호출하면 클라이언트로부터 Ack 메시지를 받아서 \_AhnHS\_VerifyResponse()를 거쳐서 다시 호출되는 동기화 과정이 정상적으로 이루어지지 않은 경우
- 확인 사항
  - 클라이언트 세션 관리 상의 동기화 문제가 발생하지 않았는지 확인
  - \_AhnHS\_MakeRequest()를 호출하지 않은 상태에서 \_AhnHS\_VerifyResponse()가 호출되지 않았는지 확인

#### **ERROR\_ANTICPXSVR\_NOT\_ENOUGH\_MEMORY (Value = 0xE9040007)**

- 설명: 메모리 부족
- 원인: 서버 메모리가 부족한 경우
- 확인사항: 서버에 메모리 leak 현상이 발생하는지 확인

#### **ERROR\_ANTICPXSVR\_BAD\_MESSAGE (Value = 0xE9040008)**

- 설명: 메시지 암복호화 실패, 패킷의 크기 혹은 형식이 정상적이지 않음.
- 원인: 검증받을 버퍼 값이 클라이언트로부터 정확히 받지 않은 경우
- 확인사항: 클라이언트에서 보낸 버퍼가 정확하게 pbyResponse로 들어왔는지 확인

#### **ERROR\_ANTICPXSVR\_REPLY\_ATTACK (Value = 0xE9040009)**

- 설명: 패킷 분석을 위한 재전송 공격 감지
- 원인: HackShield 클라이언트가 동작하지 않은 상태에서 버퍼가 전송된 경우
- 확인사항: 패킷 분석을 위한 해커의 공격일 가능성성이 있음

#### **ERROR\_ANTICPXSVR\_HSHIELD\_FILE ATTACK (Value = 0xE904000A)**

- 설명: HackShield 모듈 변조 감지

- 원인: HackShield 파일 ehsvc.dll 이 변조된 경우
- 확인사항: 클라이언트의 ehsvc.dll 파일이 변조되었는지 확인

#### **ERROR\_ANTICPXSVR\_CLIENT\_FILE\_ATTACK (Value = 0xE904000B)**

- 설명: 클라이언트 파일 변조 감지
- 원인: 게임 클라이언트 파일이 변조된 경우
- 확인사항: 게임 클라이언트가 정상적으로 업데이트가 되었는지 확인. 바이러스에 의해 클라이언트가 감염됐는지 확인

#### **ERROR\_ANTICPXSVR\_MEMORY\_ATTACK (Value = 0xE904000C)**

- 설명: 메모리 변조 감지
- 원인: 클라이언트 메모리가 변조된 경우
- 확인사항: 클라이언트 메모리를 조작하는 해킹툴이 있는지 확인

#### **ERROR\_ANTICPXSVR\_OLD\_VERSION\_CLIENT\_EXPIRED (Value = 0xE904000D)**

- 설명: 이전 버전의 클라이언트가 접속
- 원인: 서버를 종료시키지 않고 hsb 파일을 업로드했을 때, 기존에 있던 hsb 파일에 대한 클라이언트가 접속을 시도한 경우
- 확인사항: 에러가 발생한 게임 유저의 클라이언트 버전을 확인. 패치가 정상적으로 이루어지지 않아 게임에서 허용하지 않는 클라이언트 버전일 가능성이 있음

#### **참고**

HSB 파일 생성 시 HSBGen.ini에 있는 GrantOldSession을 0으로 설정한 경우 발생할 수 있습니다. 또는 GrantOldSession이 1이지만 허용할 수 있는 클라이언트 버전 개수가 MaxAllowedNumber를 초과한 경우 발생할 수도 있습니다.

예:

```
[VERCT]
GrantOldSession = 1
MaxAllowedNumber = 1
```

위 예는 현재 서버에 올라와 있는 HSB 파일에 맞는 클라이언트만 허용하겠다는 의미입니다.

#### **ERROR\_ANTICPXSVR\_UNKNOWN\_CLIENT (Value = 0xE904000E)**

- 설명: hsb 파일 생성 시 지정한 클라이언트와 짹이 맞지 않음
- 원인: HSBGen 으로 hsb 생성 시 사용된 클라이언트 버전이 아닌 경우
- 확인사항
  - 개발 과정인 경우 hsb 파일을 다시 생성하여 테스트 진행
  - 서비스 단계에서 특정 게임 유저에게 발생하는 경우, 클라이언트 파일 패치가 정상적으로 이루어지지 않았거나 바이러스에 클라이언트 파일이 감염됐는지 확인

#### **ERROR\_ANTICPXSvr\_NANOENGINE\_FILE\_ATTACK (Value = 0xE9040010)**

- 설명: 3n.mhe 파일 변조 감지
- 원인: 3n.mhe 파일이 변조된 경우
- 확인사항: 3n.mhe 파일이 변조됐는지 확인

#### **ERROR\_ANTICPXSvr\_INVALID\_HACKSHIELD\_VERSION (Value = 0xE9040011)**

- 설명: 서버에서 지원하지 않는 HackShield 버전
- 원인: 서버의 hshield.dat 보다 클라이언트의 hshield.dat 가 더 낮은 버전인 경우
- 확인사항: 접속하는 클라이언트의 hshield.dat 버전과 서버의 hshield.dat 버전이 동일한지 확인

#### **ERROR\_ANTICPXSvr\_INVALID\_ENGINE\_VERSION (Value = 0xE9040012)**

- 설명: 서버에서 지원하지 않는 휴리스틱 엔진 버전
- 원인: 서버의 3n.mhe 보다 클라이언트의 3n.mhe 가 더 낮은 버전인 경우
- 확인사항: 클라이언트 HackShield 폴더의 3n.mhe 파일이 존재하지 않거나 이전 버전인지 확인

#### **ERROR\_ANTICPXSvr\_VERIFY\_EXCEPTION (Value = 0xE9040015)**

- 설명: 예외(Exception) 발생. 단, Windows 용 모듈에서만 발생)
- 원인: 여러 가지 원인
- 확인사항: 예외가 발생한 경우, Windows 환경에서만 내부적으로 MiniDump 파일을 남기도록 되어 있음. 'VERIFYRESPONSE\_년\_월\_일 시-분-초.dmp' 형식으로 생성된 덤프 파일을 안랩 HackShield 기술 지원에 전달

#### **ERROR\_ANTICPXSvr\_ABNORMAL\_HACKSHIELD\_STATUS (Value = 0xE9040018)**

- 설명: HackShield 동작 상태 체크 결과 비정상
- 원인: 클라이언트에서 동작 중인 HackShield 에서 필요한 기능이 비정상적으로 동작하는 경우
- 확인사항: 클라이언트에서 동작 중인 HackShield 를 공격하는 해킹툴이 있는지 확인

#### **ERROR\_ANTICPXSvr\_DETECT\_CALLBACK\_IS\_NOTIFIED (Value = 0xE9040019)**

- 설명: 클라이언트에서 해킹툴 감지
- 원인: 클라이언트에서 해킹툴이 감지된 경우
- 확인사항: 클라이언트에서 동작 중인 HackShield 를 공격하는 해킹툴이 있는지 확인

#### **ERROR\_ANTICPXSvr\_ABNORMAL\_HACKSHIELD\_XSTATUS (Value = 0xE904001B)**

- 설명: HackShield 동작 상태 강화 검사 결과가 비정상
- 원인: 클라이언트에서 HackShield 가 정상적으로 동작하지 않는 경우
- 확인사항: 클라이언트에서 동작 중인 HackShield 를 공격하는 해킹 툴이 있는지 확인

#### **ERROR\_ANTICPXSvr\_OLD\_HACKSHIELD\_VERSION (Value = 0xE904001C)**

- 설명: HackShield 동작 상태 강화 검사 기능이 없는 구 버전의 HackShield 클라이언트가 접속
- 원인: HackShield 동작 상태 강화 검사 기능이 없는 구 버전의 HackShield 클라이언트가 접속했거나 구 버전의 HackShield 동작 상태를 모방한 해킹 툴이 접속한 경우(단, 구 버전의 HackShield 클라이언트의 버전은 5.5.x.x 이하를 의미)
- 확인사항: 클라이언트에서 사용 중인 HackShield 의 버전을 확인

#### **ERROR\_ANTICPXSvr\_UNKNOWN (Value = 0xE90400FF)**

- 설명: 정의되지 않은 에러
- 원인: 정상적인 경우에 발생하지 않는 에러
- 확인사항: 추가 확인 작업이 필요하기 때문에 안랩 HackShield 기술 지원에 문의

#### **! 주의**

다음의 에러 코드가 리턴된 경우, 클라이언트와의 세션을 끊어서 게임을 더 이상 진행할 수 없도록 할 것을 권장합니다. 나머지 메시지의 경우에는 고객 정책에 따라 추가할 수 있습니다.

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  
 ERROR\_ANTICPXSvr\_ABNORMAL\_HACKSHIELD\_XSTATUS  
 ERROR\_ANTICPXSvr\_OLD\_HACKSHIELD\_VERSION

#### **Example**

\_AhnHS\_VerifyResponse() 호출 예제는 다음과 같습니다.

```

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_ABNORMAL_HACKSHIELD_XSTATUS ||
        ulRet == ERROR_ANTICPXSvr_OLD_HACKSHIELD_VERSION ||
        ulRet == ERROR_ANTICPXSvr_DETECT_CALLBACK_IS_NOTIFIED )
    {
        // 예러 처리
        // Client Out -> _AhnHS_CloseClientHandle를
        // 호출하여 hClient 핸들을 닫아줌

    }
}

```

그림 25 \_AhnHS\_VerifyResponse() 호출 예제

#### 4.3.9 \_AhnHS\_MakeResponse()

클라이언트에서 사용하는 함수입니다. 서버에서 전달한 암호화된 버전 요청 메시지를 복호화하고 현재 클라이언트 파일의 버전을 암호화하여 응답 메시지를 만듭니다.

```

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

```

#### Parameters

| Parameter       | Type                | Description     |
|-----------------|---------------------|-----------------|
| pbyRequest      | unsigned char *     | [IN] 요청 메시지 버퍼  |
| ulRequestLength | unsigned long       | [IN] 요청 메시지 길이  |
| pResponseBuffer | PAHNHS_TRANS_BUFFER | [OUT] 응답 메시지 버퍼 |

#### Return Value

**ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

#### **ERR\_ANTICPXCNT\_INVALID\_PARAMETER (Value = 0xE4010001)**

- 설명: 잘못된 파라미터 값
- 원인: pbyRequest, pResponseBuffer 값이 NULL 인 경우
- 확인사항: pbyRequest, pResponseBuffer 값의 NULL 여부 확인

#### **ERR\_ANTICPXCNT\_INVALID\_ADDRESS (Value = 0xE4010002)**

- 설명: 잘못된 메모리 주소 접근
- 원인: 구조적인 문제일 가능성이 있는 경우
- 확인사항: 구조적인 문제일 수 있기 때문에 안랩 HackShield 기술 지원에 문의

#### **ERR\_ANTICPXCNT\_NOT\_ENOUGH\_MEMORY (Value = 0xE40100013)**

- 설명: 메모리 부족
- 원인: 메모리 할당이 실패한 경우
- 확인사항: 서버에서 메모리 leak 현상이 발생하는지 확인

#### **ERR\_ANTICPXCNT\_CRC\_TABLE\_INIT\_FAILED (Value = 0xE4010004)**

- 설명: 초기화 실패
- 원인: pbyRequest 버퍼 크기가 ANTICPX\_TRANS\_BUFFER\_MAX (400)을 초과한 경우
- 확인사항: HShield.h 와 라이브러리를 최신으로 적용해서 빌드했는지 확인. 서버에서 보낸 pbyRequest 버퍼가 정확하게 전달됐는지 확인

#### **ERR\_ANTICPXCNT\_BAD\_LENGTH (Value = 0xE4010005)**

- 설명: 잘못된 메시지 크기
- 원인: 전달될 메시지 버퍼의 크기가 정확하지 않은 경우
- 확인사항: HShield.h 와 라이브러리를 최신으로 적용해서 빌드했는지 확인

#### **ERR\_ANTICPXCNT\_INSUFFICIENT\_BUFFER (Value = 0xE4010006)**

- 설명: 잘못된 크기의 버퍼 전달
- 원인: Ehsvc.dll 과 HShield.lib, Hshield.h 버전이 안 맞는 경우
- 확인사항: HShield.h 와 라이브러리를 최신으로 적용해서 빌드했는지 확인

#### **ERR\_ANTICPXCNT\_NOT\_SUPPORTED (Value = 0xE4010007)**

- 설명: 현재 버전에서 지원하지 않는 기능
- 원인: 전달된 메시지 타입이 HackShield 에서 정의되지 않은 경우
- 확인사항: 서버의 HackShield 버전과 클라이언트의 HackShield 버전이 동일한지 확인

#### **ERR\_ANTICPXCNT\_FILE\_NOT\_FOUND (Value = 0xE4010008)**

- 설명: 클라이언트 파일 찾기 실패
- 원인: 클라이언트 파일을 찾을 수 없는 경우
- 확인사항: 안랩 HackShield 기술 지원에 문의

#### **ERR\_ANTICPXCNT\_INVALID\_MESSAGE\_SIZE (Value = 0xE4010009)**

- 설명: 잘못된 크기의 메시지 입력
- 원인: 서버로부터 전달받은 메시지 크기가 잘못된 경우
- 확인사항: 클라이언트의 HackShield 버전과 서버의 HackShield 버전이 동일한지 확인

#### **ERR\_ANTICPXCNT\_BAD\_FORMAT (Value = 0xE401000A)**

- 설명: 잘못된 포맷
- 원인: 서버로부터 전달받은 메시지 크기가 잘못된 경우
- 확인사항: 클라이언트의 HackShield 버전과 서버의 HackShield 버전이 동일한지 확인

#### **ERR\_ANTICPXCNT\_DEBUGGER\_DETECTED (Value = 0xE401000B)**

- 설명: 디버그 상황 감지
- 원인: 해당 클라이언트가 디버깅 상태인 경우
- 확인사항: 디버거나 현재 디버깅 중이 아닌지 확인

#### **ERR\_ANTICPXCNT\_BAD\_HSHIELD\_MODULE (Value = 0xE401000C)**

- 설명: 잘못된 HackShield 모듈 경로나 잘못된 HackShield 모듈
- 원인: HackShield 모듈이 존재하지 않는 경우
- 확인사항: HackShield 모듈 경로 및 HackShield 모듈이 올바른지 확인

#### **ERR\_ANTICPXCNT\_BAD\_CLIENT\_FILE (Value = 0xE401000D)**

- 설명: 잘못된 클라이언트 모듈
- 원인: 클라이언트 파일 접근 시 문제가 발생한 경우
- 확인사항: N/A

#### **ERR\_ANTICPXCNT\_BAD\_REQUEST (Value = 0xE401000E)**

- 설명: 잘못된 요청 메시지를 서버가 전달
- 원인: 서버로 받은 버퍼가 정상적으로 받지 못해 복호화 실패 메시지가 발생한 경우
- 확인사항: 서버로부터 전송된 버퍼의 값과 길이가 정확하게 pbyRequest로 들어왔는지 확인

#### **ERR\_ANTICPXCNT\_HSHIELD\_CORE\_ENGINE\_NOT\_WORKING (Value = 0xE401000F)**

- 설명: HackShield 코어 엔진의 비정상적인 동작
- 원인: 정상적인 경우에 발생하지 않는 에러
- 확인사항: 추가 확인 작업이 필요하기 때문에 안랩 HackShield 기술 지원에 문의

#### **ERR\_ANTICPXCNT\_UNKNOWN (Value = 0xE40100FF)**

- 설명: 해킹 시도로 시스템이 오동작
- 원인: 정상적인 경우에 발생하지 않는 에러
- 확인사항: 추가 확인 작업이 필요하기 때문에 안랩 HackShield 기술 지원에 문의

#### **Example**

`_AhnHS_MakeResponse()` 사용 예제는 다음과 같습니다.

```
ulRet = _AhnHS_MakeResponse ( stRequestBuf.byBuffer,
                               stRequestBuf.nLength,
                               &stResponseBuf ) ;

if ( ulRet != ERROR_SUCCESS )
{
    // 에러 처리
}
```

**그림 26 `_AhnHS_MakeResponse()` 사용 예제**

# **5장**

## **모니터링 서비스 기능**

- 5.1 모니터링 서비스 기능 /127**
- 5.2 애플리케이션 프로그래밍 /129**
- 5.3 애플리케이션 프로그래밍 인터페이스 /131**

## 5.1 모니터링 서비스 기능

---

HackShield의 모니터링 서비스 기능을 소개하고 이 기능을 구현하기 위한 방법을 설명합니다.

### 5.1.1 기능 소개

모니터링 서비스는 HackShield가 적용된 프로그램의 해킹 상황과 에러 상황을 중앙에서 모니터링하는 프로그램입니다. 해킹의 확산과 에러로 인한 장애는 전체 서비스에 큰 피해를 입힐 수 있기 때문에 해킹 및 에러 상황을 실시간으로 모니터링하여 초기에 대응하도록 합니다. 또한 보고서 기능을 제공함으로써, 서비스 상황의 통계 자료로서 매우 유용한 기능을 제공합니다. 모니터링 서비스 기능은 다음과 같습니다.

- 실시간 해킹/에러 모니터링
- 보고서
- 정책 관리

#### 실시간 해킹/에러 모니터링

HackShield가 적용된 프로그램의 해킹 상황과 에러 상황에 대해 HackShield가 전달한 정보를 실시간으로 모니터링할 수 있습니다

#### 보고서

수집된 로그로 다양하게 통계를 내어 필요한 정보를 보고서로 출력하여 볼 수 있습니다.

#### 정책 관리

필요한 상황에 맞게 다양한 정책을 설정하고 DB 정보를 백업할 수 있습니다.

### 5.1.2 기능 특징

모니터링 서비스 기능의 특징은 다음과 같습니다.

#### 인터페이스 함수(API) 제공

Ehsvc에서 제공하는 기능을 편리하게 사용하고 그 실행 결과 값을 받아 볼 수 있도록 인터페이스 DLL과 라이브러리를 제공합니다. 게임 개발자는 인터페이스 DLL과 라이브러리를 사용하여 고객 고유 정책에 따라 게임 파일의 무결성 및 HackShield의 정상 동작 유무를 확인할 수 있습니다.

## 모니터링 서버 프로그램 제공

게임 클라이언트에서 보내온 해킹 및 에러 정보를 실시간으로 확인할 수 있도록 오라클을 사용하여 DB를 관리하고 실시간으로 확인할 수 있는 모니터링 프로그램을 제공합니다.

## 테스트 프로그램 제공

Ehsvc에서 제공하는 API를 사용하여 구현한 테스트용 프로그램인 Amazon.exe를 제공합니다. Amazon.exe는 기존의 HackShield 테스트 기능과 Ehsvc의 테스트 기능을 제공합니다.

### 5.1.3 시스템 구조

HShield.lib, EHService.dll을 제공하여 서버와 클라이언트에 DLL과 라이브러리 형태로 적용합니다. 모니터링 서비스의 전체 구조 및 동작 원리는 다음과 같습니다.

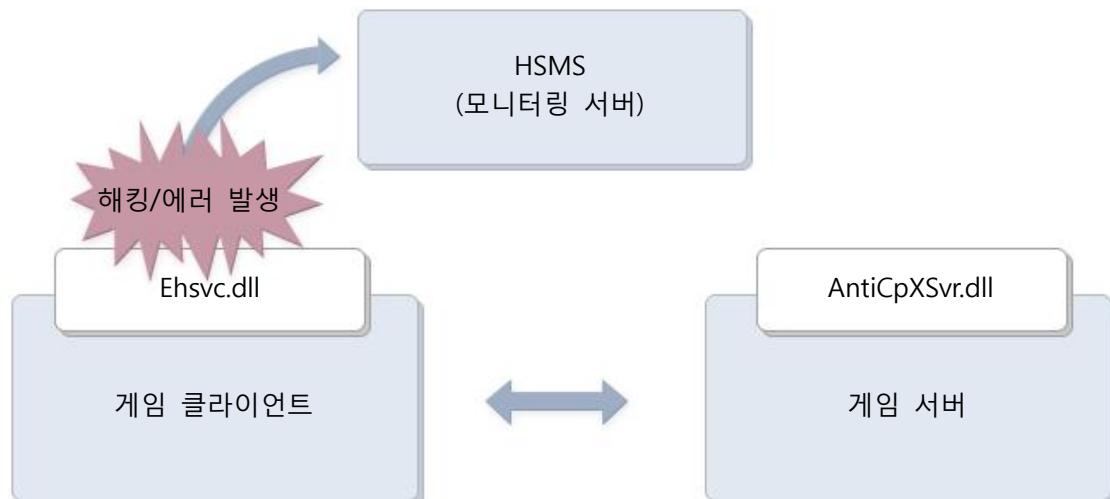


그림 27 모니터링 서비스의 동작 원리

위 그림에서 보면 게임 클라이언트에서 `_AhnHS_StartMonitor()` 호출하면, 해당 클라이언트에서 발생한 해킹 및 에러 정보를 모니터링 서버에 전송합니다.

모니터링 서버에서는 클라이언트 프로그램을 특정 게임 코드로 관리합니다. 또한 해당 게임 코드와 관련해서 클라이언트가 전송한 정보를 DB로 관리하고 웹 상에서 쉽게 액세스 할 수 있습니다.

위 그림의 주요 항목에 대한 설명은 다음과 같습니다.

### Ehsvc.dll (인터페이스 dll)

해킹 및 에러가 발생한 경우 모니터링 서버에 알릴 수 있도록 기본적인 정보를 세팅하는 API입니다.

## HShield.lib (HackShield 라이브러리)

해킹 및 에러가 발생한 경우 모니터링 서버에 알릴 수 있도록 기본적인 정보를 세팅하는 API입니다.

## HSMS\_Setup.exe(모니터링 서버 사이드 설치 파일)

게임 클라이언트에서 보낸 정보를 DB로 관리하고 웹 상에서 쉽게 관리할 수 있게 돋는 서버 사이드 프로그램입니다.

## 5.2 애플리케이션 프로그래밍

Ehsvc에서 제공한 API를 이용해서 모니터링 서비스 기능을 구현하는 방법을 설명합니다.

### 참고

이 장에서 사용하고 있는 샘플 코드는 Microsoft Visual C++ 6.0을 기준으로 한 C/C++ 언어로 작성했습니다. 프로그래밍에 사용되는 언어는 각 프로그램의 특성 및 시스템 환경에 따라서 변경할 수 있습니다.

### 5.2.1 프로그래밍 준비

모니터링을 사용하여 프로그래밍을 시작하기 전에 확인해야 하는 모니터링 관련 파일 목록은 다음과 같습니다.

표 17 모니터링 관련 파일

| 파일 이름       | 설치 폴더        | 설명                                       |
|-------------|--------------|------------------------------------------|
| HShield.h   | [프로그램 소스 폴더] | 클라이언트에서 사용할 헤더 파일<br>HackShield 기능 포함    |
| HShield.lib | [프로그램 소스 폴더] | 클라이언트에서 사용할 라이브러리 파일<br>HackShield 기능 포함 |
| EHSvc.dll   | [프로그램 소스 폴더] | 클라이언트에서 사용할 DLL 파일<br>HackShield 기능 포함   |

### 5.2.2 프로그래밍 적용

프로그래밍을 하기 위해 서버와 클라이언트를 준비하고 적용하는 방법은 다음과 같습니다.

#### 서버 적용 방법

서버 적용 방법은 다음과 같습니다.

1. HackShield 모니터링 서버 설치 가이드를 참고하여 해당 서버에 HackShield 모니터링 서버 프로그램을 설치합니다.

- HackShield 모니터링 서버 운용 가이드를 참고하여 운용하십시오.

## 클라이언트 적용 방법

클라이언트 적용 방법은 다음과 같습니다.

- HShield.lib 를 작업할 프로젝트에 포함시킵니다.
- 제공되는 HShield.h 를 작업할 소스 파일에 포함시킵니다.
- HShield.h 에 정의된 AHNHS\_EXT\_ERRORINFO 구조체에 대한 변수를 생성하고 초기화합니다.

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

- HsExtError 구조체 내의 szServer(모니터링 서버 주소), szUserId(게임 유저 계정), szGameVersion(게임 버전)에 정보를 입력합니다.

```
HsExtError.szServer = "127.0.0.1"           // 모니터링 주소  
HsExtError.szUserId = "Test"                 // 게임 유저 ID  
HsExtError.szGameVersion = "3.0.0.1"        // 게임 버전
```

---

### !

#### 참고

모니터링 IP와 포트 정보는 HSUpdate.env를 통해 추후 수정이 가능합니다.

---

### !

#### 주의

모니터링 기능을 사용하는 경우에는 HSUpdate.env파일에 유효한 모니터링 IP와 포트 정보가 삽입이 되었는지를 확인해 주십시오. 그와 반대로 모니터링 기능을 사용하지 않는 경우에는 HSUpdate.env파일에 모니터링 IP와 포트 정보가 없는지를 확인해야 합니다.

- 4 단계에서 생성한 구조체 변수 값과 Ehsvc.dll 의 전체 경로를 인자 값으로 전달하여 \_AhnHS\_StartMonitor()를 호출합니다.

---

### !

#### 주의

\_AhnHS\_StartMonitor()는 \_AhnHS\_Initialize() 이전에 반드시 호출하십시오. 그렇지 않으면 HackShield 구동 전 발생한 에러가 전송되지 않습니다.

또한 HackShield 업데이트 라이브러리와 HackShield 라이브러리를 동일한 게임 모듈에서 적용하는 경우에는 반드시 HackShield 업데이트 라이브러리를 적용한 다음 호출하십시오. 그렇지 않으면 HackShield 업데이트를 정상적으로 할 수 없습니다.

```
lstrcat (szFullFileName, _T("\\\\HShield\\\\Ehsvc.dll"));

dwRet = _AhnHS_StartMonitor ( HsExtError      // [in]
                            szFullFileName // [in]
                           );
if( dwRet != ERROR_SUCCESS )
{
```

```

        // 실패가 발생한다면 모니터링 기능만 동작이 안 되는 것으로
        // 예전에 관련 로그만 기록되도록 합니다.
    }

dwRet = _AhnHS_Initialize ( ...

```

6. 4 단계에서 게임 유저 ID 정보를 얻어올 수 없는 시점인 경우, 게임 유저 ID 를 얻어올 수 있는 시점에 `_AhnHS_SetUserId()` 혹은 `_AhnHS_UpdateMonitorInfo()`를 호출하면 게임 유저 ID 를 세팅할 수 있습니다.

## 5.3 애플리케이션 프로그래밍 인터페이스

HackShield의 모니터링 서비스 기능 구현에 필요한 API를 소개합니다.

### 5.3.1 `_AhnHS_StartMonitor()`

해킹 및 에러 발생 시 데이터를 전송할 서버 정보를 세팅하는 함수입니다.  
`_AhnHS_Initialize()` 이전에 호출합니다.

```

int
__sdccall
_AhnHS_StartMonitor (IN AHNHS_EXT_ERRORINFO HsExtErrorInfo,
                     IN LPCSTR szFileName);

```

#### Parameters

| Parameter      | Value               | Description                               |
|----------------|---------------------|-------------------------------------------|
| HsExtErrorInfo | AHNHS_EXT_ERRORINFO | 서버 URL 주소, 게임 유저 ID, 게임 버전 정보를 가지고 있는 구조체 |
| szFileName     | LPCSTR              | Ehsvc.dll 전체 경로                           |

#### Return Value

**ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 함수 호출 성공
- 원인: 정상적인 경우
- 확인사항: N/A

**HS\_ERR\_INVALID\_FILES (Value = 0x1C001)**

- 설명: 잘못된 입력 값
- 원인: 파라미터 값이 NULL 인 경우
- 확인사항: HsExtErrorInfo, szFileName 값의 NULL 여부 확인

### **HS\_ERR\_UNKNOWN (Value = 0x1C002)**

- 설명: 알 수 없는 에러
- 원인: 함수 내부에서 예외가 발생했거나 함수 구조적인 문제일 가능성이 큰 경우
- 확인사항: HShield.log 와 수집한 AhnReport 를 안랩 HackShield 기술 지원에 전달

## **Example**

`_AhnHS_StartMonitor()` 호출 예제는 다음과 같습니다.

```

AHNHS_EXT_ERRORINFO HsExtError;           // HShield.h에 정의된 구조체

HsExtError.szServer = "127.0.0.1"          // 모니터링 주소
// port를 지정하지 않을시 default로 7999 port를 사용하며,
// default port가 아닌 port를 이용시 아래와 같이 입력을 하시면 됩니다.
// HsExtError.szServer = ,127.0.0.1:6000'    // 6000port 사용시
HsExtError.szUserId = "Test"                // 게임 유저 ID
HsExtError.szGameVersion = "3.0.0.1"        // 게임 버전

lstrcat (szFullFileName, _T("\\\\HShield\\\\Ehsvc.dll"));

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

if( dwRet != ERROR_SUCCESS )
{
    // 실패가 발생한다면 모니터링 기능만 동작이 안되는 것이므로
    // 에러에 관한 로그만 기록되도록 합니다.
}

dwRet = _AhnHS_Initialize ( ...

```

**그림 28 `_AhnHS_StartMonitor()` 호출 예제**

### **5.3.2 `_AhnHS_SetUserId()`**

모니터링 서버에 전송될 메시지의 게임 유저 ID를 저장하는 함수입니다. `_AhnHS_StartMonitor()`에서도 게임 유저 ID 정보를 받지만 HackShield 초기화 시점에는 게임 유저 정보를 알지 못하는 경우가 있습니다. 따라서 게임 유저 ID를 아는 시점에 이 함수를 호출하여 유저 정보를 얻을 수 있습니다. 게임 유저 ID를 얻지 못해 정보가 없는 동안은 에러 값이 전송됩니다.

---

**⚠ 주의**

\_AhnHS\_SetUserId API의 경우 AhnHS\_StartMonitor() 호출 이후에 호출이 되어야 합니다

---

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

**Parameters**

| Parameter | Value  | Description                                                       |
|-----------|--------|-------------------------------------------------------------------|
| szUserID  | LPCSTR | 게임 클라이언트의 게임 유저 정보<br>최대 120 바이트까지만 지원<br>예: 영문 128 글자, 다국어 40 글자 |

**Return Value**

N/A

**Example**

\_AhnHS\_SetUserId() 호출 예제는 다음과 같습니다.

```
    _AhnHS_SetUserId ( szUserID );
```

그림 29 \_AhnHS\_SetUserId() 호출 예제

### 5.3.3 \_AhnHS\_SetUserCustomInfo()

HackShield가 해킹 및 에러 정보를 감지했을 때 해당 게임 유저의 정보를 모니터링 서버로 전송하는 함수입니다. 단, 게임에서 구현된 HackShield 콜백 함수 내부에서는 이 함수를 호출해도 모니터링 서버로 전송되지 않습니다.

```
int  
__stdcall  
_AhnHS_SetUserCustomInfo (  
    IN const char* szUserCustomInfo  
);
```

**Parameters**

| Parameter        | Value       | Description                              |
|------------------|-------------|------------------------------------------|
| szUserCustomInfo | const char* | 모니터링 서버로 보낼 게임 유저 정보<br>최대 512 바이트까지만 지원 |

## Return Value

### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우. 해킹툴 감지 등으로 HackShield 가 모니터링 서버로 정보 전송 시 추가 정보도 함께 전송

### HS\_ERR\_INVALID\_PARAM (Value = 0x002)

- 설명: 잘못된 파라미터
- 원인: szUserCustomInfo 값이 NULL 인 경우

## Example

\_AhnHS\_SetUserCustomInfo() 호출 예제는 다음과 같습니다.

```
CHAR szUserCustomInfo[512] = { 0, };
StringCchcopy( szUserCustomInfo, "Extra Data : 1234 ");
_AhnHS_SetUserCustomInfo ( szUserCustomInfo );
```

그림 30 \_AhnHS\_SetUserCustomInfo() 호출 예제

### 5.3.4 \_AhnHS\_SendUserCustomInfo()

파라미터를 통해 입력된 게임 유저 정보를 모니터링 서버로 전송하는 함수입니다.

```
int
__stdcall
_AhnHS_SendUserCustomInfo (
    IN const char* szUserCustomInfo,
    IN DWORD dwTimeout = DEFAULT_HSMS_TIME_OUT
);
```

## Parameters

| Parameter        | Value       | Description                                                                                         |
|------------------|-------------|-----------------------------------------------------------------------------------------------------|
| szUserCustomInfo | const char* | 사용자 정의 정보<br>최대 512 바이트까지만 지원                                                                       |
| dwTimeout        | DWORD       | 사용자 정의 정보를 모니터링 서버로<br>전송 완료하기까지 대기하는 시간<br>DEFAULT_HSMS_TIME_OUT = 5000(약 5초)<br>0인 경우 대기 시간 없이 리턴 |

## Return Value

### HS\_ERR\_OK (Value = 0x000)

- 설명: 함수 호출 성공
- 원인: 정상적인 경우. 해킹툴 감지 등으로 HackShield 가 모니터링 서버로 정보 전송 시 추가 정보도 함께 전송
- 확인사항: N/A

#### **HS\_ERR\_UNKNOWN (Value = 0x001)**

- 설명: 알 수 없는 에러
- 원인: 함수 내부에서 예외가 발생했거나 함수 구조적인 문제일 가능성이 큰 경우
- 확인사항: HShield.log 와 수집한 AhnReport 를 안랩 HackShield 기술 지원에 전달

#### **HS\_ERR\_INVALID\_FILES (Value = 0x101)**

- 설명: HackShield 가 초기화되지 않음
- 원인: \_AhnHS\_Initialize() 를 호출하지 않았거나 함수 호출에 실패하여 HackShield 를 초기화하지 않은 상태에서 이 함수를 호출한 경우
- 확인사항: HackShield 가 시작되지 않았거나 해킹으로 인해 HackShield 가 정상 동작하지 않음

#### **HS\_ERR\_INVALID\_PARAM (Value = 0x002)**

- 설명: 잘못된 파라미터
- 원인: AhnHS\_StartService 및 모니터링 서비스가 시작되지 않았을 경우
- 확인사항: 모니터링 서비스가 정상적으로 발송되지 않음

#### **HS\_ERR\_HSMS\_WAIT\_TIME\_OUT (Value = 0x801)**

- 설명: 설정된 시간 안에 에러정보를 모니터링 서버로 전송하지 못함
- 원인: 설정된 시간이 너무 짧거나 에러 정보 전송 시 문제가 발생되어 전송하지 못한 경우
- 확인사항: N/A

#### **HS\_ERR\_HSMS\_NOT\_RUNNING (Value = 0x803)**

- 설명: HackShield 모니터링 서비스가 시작하지 않음
- 원인: HackShield 모니터링 서비스가 시작하지 않은 상태에서 호출한 경우
- 확인사항: 개발 과정에서만 발생하는 에러이므로 별도 처리 필요 없음

### **Remarks**

해당 함수 적용 시에 HackShield 내부의 로직에 의해 최소 5초 이상의 타임아웃 값을 필요로 합니다. 만약 타임아웃 값이 0초면 대기 시간 없이 리턴합니다. \_AhnHS\_StartMonitor(), \_AhnHS\_StartService() 후에 호출해야 하며, 모니터링 서비스에는 사용자 정의 에러 코드(0xFFFF0001)로 전송됩니다.

## Example

\_AhnHS\_SendUserCustomInfo() 호출 예제는 다음과 같습니다.

```
// 동기 호출 - 타임 아웃 5초  
_AhnHS_SendUserCustomInfo ( "Param" );  
  
// 대기시간 없음  
_AhnHS_SendUserCustomInfo ( "Param", 0 );  
  
// 동기 호출 - 타임 아웃 50초  
_AhnHS_SendUserCustomInfo ( "Param", 50000 );
```

그림 31 \_AhnHS\_SendUserCustomInfo() 호출 예제

### 5.3.5 \_AhnHS\_UpdateMonitorInfo()

모니터링 서버에 전송될 메시지에 함께 전달되는 정보들을 모니터링 기능을 사용중에 변경할 수 있도록 하는 함수입니다. 현재는 게임 유저 ID 및 게임 버전 정보만이 해당 함수를 통해 변경될 수 있는 정보이지만, 추후 전달되는 정보들의 확장성을 고려하여 고안된 함수입니다. 물론, \_AhnHS\_SetUserID()에서도 게임 유저 ID 정보를 변경할 수 있습니다. 단, \_AhnHS\_SetUserID()를 통해 게임 유저 ID를 변경한 이후에 다시 \_AhnHS\_UpdateMonitorInfo()를 호출하면 \_AhnHS\_UpdateMonitorInfo()에 전달된 게임 유저 ID로 변경됩니다.

```
void  
__sdccall  
_AhnHS_UpdateMonitorInfo (IN AHNHS_EXT_ERRORINFO HsExtErrorInfo);
```

#### Parameters

| Parameter      | Value               | Description                                      |
|----------------|---------------------|--------------------------------------------------|
| HsExtErrorInfo | AHNHS_EXT_ERRORINFO | 서버 URL 주소(=NULL), 게임 유저 ID, 게임 버전 정보를 가지고 있는 구조체 |

#### Return Value

N/A

#### Remarks

서버 URL 주소에 해당하는 HsExtErrorInfo.szServer 는 NULL로 처리합니다. NULL이 아닌 값을 넣으면, 해당 값은 무시됩니다. 즉, 앞서 호출되었던 \_AhnHS\_StartMonitor() 함수를 통해 전달되었던 HsExtErrorInfo.szServer 값을 그대로 유지합니다.

## Example

\_AhnHS\_UpdateMonitorInfo() 호출 예제는 다음과 같습니다.

```
AHNHS_EXT_ERRORINFO HsExtError;           // HShield.h에 정의된 구조체

memset(HsExtError.szServer, 0x0, sizeof(HsExtError.szServer)); // 모니터링주소
_stprintf(HsExtError.szUserId, "UserID");           // 게임 유저 ID
_stprintf(HsExtError.szGameVersion, "3.0.0.1"); // 게임 버전

_AhnHS_UpdateMonitorInfo (HsExtError );           // [in]
```

그림 32 \_AhnHS\_UpdateMonitorInfo() 호출 예제

# **6장**

# **LMP 기능**

**6.1 Local Memory Protection(LMP) 기능 /139**

**6.2 애플리케이션 프로그래밍 /141**

**6.3 애플리케이션 프로그래밍 인터페이스 /144**

## 6.1 Local Memory Protection(LMP) 기능

HackShield의 LMP 기능을 소개하고 이 기능을 구현하기 위한 방법을 설명합니다.

### 6.1.1 기능 소개

LMP 기능은 일부 패커들을 사용했을 때, 기존의 서버 연동 기능의 메모리 보호 방식으로 보호할 수 없는 경우를 대비하여 클라이언트 단에서 메모리 위/변조를 감지합니다. 기존의 서버 연동에서의 메모리 변조 감지 기능과 유사하지만 서버를 거치지 않고 클라이언트 자체적으로 자신의 메모리 영역을 보호할 수 있게 합니다.

#### !

#### 참고

Themida 패커에서 메모리 보호를 위해 매번 실행될 때마다 서로 다른 메모리 주소의 코드를 변경 합니다. 따라서 서버에서 메모리 CRC를 체크하는 서버 연동 방식과 충돌이 발생합니다.

LMP의 기능을 정리하면 다음과 같습니다.

- HSBGen 툴을 이용하여 EXE이나 DLL에 보호 파일 적용 가능
- THEMIDA의 API Wrapping을 지원하는 로컬 메모리 보호 기능
- 서버 연동의 hsb 파일 생성기로 게임 클라이언트 파일에 적용 가능
  - 보호 파일: EXE, DLL 파일
  - 사용툴: HSBGen.exe

### 실행 파일의 메모리 보호

실행 파일의 메모리에서 코드 영역에 대한 위/변조를 감지합니다.

### DLL 파일의 메모리 보호

지정한 DLL 파일의 코드 영역에 대해서도 위/변조를 감지합니다. 해당 DLL이 직접 빌드하지 않는 파일이어도 보호가 가능합니다.

#### !

#### 주의

동적으로 로딩되는 DLL 파일을 보호할 경우 \_AhnHS\_Initialize(), \_AhnHS\_StartService() 호출 시점 후에 해당 DLL을 로딩한다면, DLL 로딩 후에 반드시 \_AhnHS\_IsModuleSecure(szDllPath)()를 호출하십시오.

### 메모리 무결성 검증

LMP 기능이 동작하기 전에 메모리를 수정하는 행위를 방지하기 위해 일부 시스템 파일의 후킹 여부를 조사하고 메모리에 대해 무결성을 검증합니다.

### 6.1.2 기능 특징

LMP 기능의 특징은 다음과 같습니다.

#### 옵션 제공

초기화 부분에서 AHNHS\_CHKOPT\_LOCAL\_MEMORY\_PROTECTION 옵션을 적용을 하면 해당 기능이 실행이 됩니다.

#### 콜백 메시지 제공

메모리 조작이 감지되면 HackShield 콜백 함수에서 AHNHS\_ACTAPC\_DETECT\_MEM MODIFY\_FROM\_LMP 또는 AHNHS\_ACTAPC\_DETECT\_RMEM MODIFY\_FROM\_LMP 메시지가 전달됩니다.

#### 테스트 프로그램 제공

Ehsvc에서 제공하는 API를 사용하여 구현한 테스트용 프로그램인 Amazon.exe를 제공합니다. Amazon.exe는 기존의 HackShield 테스트 기능과 Ehsvc의 테스트 기능을 제공합니다.

### 6.1.3 시스템 구조

HSBGen.exe를 이용하여 보호하려는 모듈에 섹션 정보를 입력하고 HShield.lib, EHService.dll을 제공하여 클라이언트에 DLL과 라이브러리 형태로 적용됩니다.

LMP의 전체 구조 및 동작 원리는 다음과 같습니다.

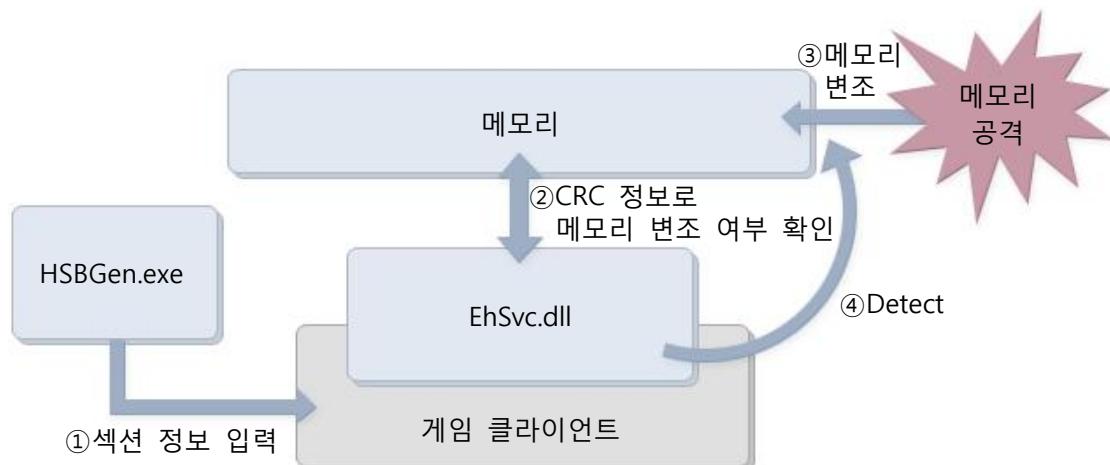


그림 33 Local Memory Protection 동작 구조

위 그림의 각 동작 구조에 대해 설명하면 다음과 같습니다.

①HackShield에서 제공하는 HSBGen.exe 툴을 이용하여 게임 클라이언트의 PE 구조 특정 영역에 게임 클라이언트나 DLL의 섹션 정보를 저장합니다.

② HackShield가 실행되면 게임 클라이언트나 DLL의 PE 구조에서 섹션 정보를 로딩하고 실행 이전에 메모리가 조작되었는지 여부를 체크합니다. 또한 섹션 정보를 바탕으로 현재 메모리의 CRC 정보를 구성하며, CRC 정보를 바탕으로 메모리 조작 여부를 주기적으로 체크합니다.

③ 메모리를 변조하는 공격이 발생합니다.

④ 주기적으로 메모리 조작 여부를 체크하여 탐지합니다.

위 그림의 주요 항목에 대한 설명은 다음과 같습니다.

### Ehsvc.dll(인터페이스 dll)

해킹 및 에러가 발생한 경우 모니터링 서버에 알릴 수 있도록 기본적인 정보를 세팅하는 API를 제공합니다.

### HShield.lib(HackShield 라이브러리)

해킹 및 에러가 발생한 경우 모니터링 서버에 알릴 수 있도록 기본적인 정보를 세팅하는 API를 제공합니다.

### HSBGen.exe(보호 모듈 설정 유틸)

보호하려는 클라이언트 파일에 섹션 정보를 입력하여 HackShield에서 해당 모듈에 대한 코드 영역을 보호하게 합니다.

## 6.2 애플리케이션 프로그래밍

Ehsvc에서 제공하는 HackShield 초기화 함수 \_AhnHS\_Initialize()를 이용해서 LMP 기능을 구현하는 방법을 설명합니다.

### 참고

이 문서에서 사용하고 있는 샘플 코드는 Microsoft Visual C++ 6.0을 기준으로 한 C/C++ 언어로 작성했습니다. 프로그래밍에 사용되는 언어는 각 프로그램의 특성 및 시스템 환경에 따라서 변경 할 수 있습니다.

### 6.2.1 프로그래밍 준비

LMP 기능을 사용하여 프로그래밍을 시작하기 전에 확인해야 하는 LMP 관련 파일 목록은 다음과 같습니다.

표 18 LMP 관련 파일

| 경로 | 파일 이름 | 설치 폴더 | 설명 |
|----|-------|-------|----|
|----|-------|-------|----|

|                              |             |              |                      |
|------------------------------|-------------|--------------|----------------------|
| [SDK]\Include\               | HShield.h   | [프로그램 소스 폴더] | 클라이언트에서 사용할 헤더 파일    |
| [SDK]\Lib                    | HSheild.lib | [프로그램 소스 폴더] | 클라이언트에서 사용할 라이브러리 파일 |
| [SDK]\Bin\Win\x86\AntiCrack\ | HSBGen.exe  |              | 보호하려는 모듈에 설정하는 유틸리티  |

## 6.2.2 프로그래밍 적용

프로그래밍을 하기 위해 클라이언트를 준비하고 적용하는 방법과 보호 모듈을 적용하는 방법, 부가 기능을 적용하는 방법은 다음과 같습니다.

### 클라이언트 적용 방법

클라이언트 적용 방법은 다음과 같습니다.

1. HShield.lib 를 작업할 프로젝트에 포함시킵니다
2. 제공되는 HShield.h 를 작업할 소스 파일에 포함시킵니다.
3. 해당 소스에 \_AhnHS\_Initialize() 호출 시 5 번째 파라미터에 다음과 같은 옵션을 추가 합니다.

```
// _AhnHS_Initialize() 호출에 쓰일 옵션 플래그를 정의합니다(기존의 옵션에 추가)
dwOption = AHNHS_CHKOPT_ALL | 
AHNHS_CHKOPT_LOCAL_MEMORY_PROTECTION;

// _AhnHS_Initialize()를 호출하여 HackShield를 초기화합니다.
nRet = _AhnHS_Initialize (   szFullPath,
                           HS_CallbackProc,           // 콜백 함수
                           1000,                      // 게임 코드
                           "B228F291B7D7FAD361D7A4B7", // 라이선스 키
                           dwOption,                  // 옵션 플래그
                           AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL);
...
```

4. HackShield 와 관련하여 이벤트 전달 함수 작성 부분에

AHNHS\_ACTAPC\_DETECT\_MEM MODIFY\_FROM\_LMP 와  
AHNHS\_ACTAPC\_DETECT\_RMEM MODIFY\_FROM\_LMP 처리를 추가합니다.

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

    // 각 경우에 대하여 알맞은 에러 메시지를 출력합니다
    switch ( lCode )
    {
        //
        // LMP 콜백
        // 참고: LMP 콜백에서는 변조된 모듈 이름과 페이지상의 주소 값도
```

```

// 함께 넘어오지만 이를 사용자에게 직접 노출할 필요는 없습니다.
case AHNHS_ACTAPC_DETECT_MEM MODIFY_FROM_LMP:
    wsprintf(szMsg, "모듈에서 메모리
변조가 감지되었습니다.\n" );
    MessageBox( NULL, szMsg, szTitle, MB_OK );
    break;

case AHNHS_ACTAPC_DETECT_RMEM MODIFY_FROM_LMP:
    wsprintf(szMsg, "모듈에서 읽기전용 메모리
변조가 감지되었습니다.\n" );
    MessageBox( NULL, szMsg, szTitle, MB_OK );
    break;

...
}

```

파라미터로 전달되는 내용은 다음과 같습니다.

- ICode AHNHS\_ACTAPC\_DETECT\_MEM MODIFY\_FROM\_LMP(0x10705)
  - pParam
    - 조작된 모듈 이름(모듈 베이스 주소): 조작된 실제 페이지 주소
    - 파라미터로 조작이 발생한 모듈의 이름과 페이지 주소 값이 넘어옴
    - 해당 정보는 조작 발생 시 정보로 활용하며 게임 유저에게 보여줄 필요 없음
- ICode AHNHS\_ACTAPC\_DETECT\_RMEM MODIFY\_FROM\_LMP(0x10709)
  - pParam
    - 조작된 모듈 주소: 조작된 실제 페이지 주소
    - 파라미터로 조작이 발생한 모듈의 페이지 주소 값이 넘어옴
    - 해당 정보는 조작 발생 시 정보로 활용하며 게임 유저에게 보여줄 필요 없음

## 보호 모듈 적용 방법 - HSBGen.exe

해당 모듈을 패킹해서 배포하는 경우 HSBGen.exe가 적용된 후에 패커를 적용합니다.



### 참고

자세한 내용은 [9.1 HSBGen 툴](#)을 참고하십시오.

## 부가 기능 적용 방법

### \_AhnHS\_IsModuleSecure()

HSBGen.exe를 이용하여 정보를 삽입한 보호 대상 DLL 내에 삽입된 정보가 정상적으로 존재하는지 무결성 유무를 확인하는 함수입니다.

\_AhnHS\_IsModuleSecure() 호출 예제는 다음과 같습니다.

```
_AhnHS_Initialize (...);                                // HackShield 초기화  
_AhnHS_StartService (...);                            // HackShield 시작  
  
LoadLibrary("C:\\\\GAME\\\\GameEngine.dll");      // Dll-loading  
  
// 반드시 HackShield가 정상적으로 초기화되고, 보호대상 DLL이 로딩된 이후에 호출합니다.  
BOOL bRet = _AhnHS_IsModuleSecure ("C:\\\\GAME\\\\GameEngine.dll");
```

그림 34 \_AhnHS\_IsModuleSecure() 호출 예제

위 예제에서 보면 bRet이 TRUE인 경우 HSBGen 삽입 정보가 정상적으로 존재하는 것입니다. FALSE인 경우에는 HSBGen 삽입 정보가 변조됐거나 존재하지 않는 것입니다. 해당 기능은 보호 대상 dll을 Dummy 형태로 교체하는 유형의 해킹을 감지합니다.

## 6.3 애플리케이션 프로그래밍 인터페이스

HackShield의 LMP 기능 구현에 필요한 API를 소개합니다.

### 참고

이 장에서 사용하고 있는 샘플 코드는 Microsoft Visual C++ 6.0을 기준으로 한 C/C++ 언어로 작성했습니다. 프로그래밍에 사용되는 언어는 각 프로그램의 특성 및 시스템 환경에 따라서 변경할 수 있습니다.

### 6.3.1 \_AhnHS\_IsModuleSecure()

HSBGen.exe를 통해 적용된 DLL이나 EXE에, LMP 기능이 정상적으로 적용됐는지 확인하거나 적용된 LMP 데이터가 손상됐는지 확인할 때 사용하는 함수입니다.

동적으로 로드되는, 즉 LMP가 적용된 DLL에 대해 보호 요청을 할 때 사용됩니다. 정적으로 바인딩된, 즉 LMP가 적용된 DLL에 대해서는 호출할 필요가 없습니다.

```
BOOL __stdcall _AhnHS_IsModuleSecure (IN LPCSTR szModulePath )
```

#### Parameters

| Parameter    | Value  | Description         |
|--------------|--------|---------------------|
| szModulePath | LPCSTR | DLL이나 EXE 경로(전체 경로) |

#### Return Value

TRUE

- 설명: 모듈의 LMP 정보가 유효하며 정상적으로 LMP 보호 목록 추가
- 원인: 정상적인 경우
- 확인사항: N/A

### FALSE

- 설명: 전달된 경로의 모듈에 LMP 정보가 없거나 유효하지 않으므로 LMP에 의해 보호되지 않음
- 원인: 전달된 경로의 모듈이 다른 모듈로 교체됐거나 LMP 정보가 손상된 경우
- 확인사항: 전달된 경로의 모듈에 LMP 가 정상적으로 적용됐는지 확인

### Example

`_AhnHS_IsModuleSecure()` 호출 예제는 다음과 같습니다.

```
BOOL bRet = _AhnHS_IsModuleSecure ("C:\\GAME\\GameEngine.dll");
```

그림 35 `_AhnHS_IsModuleSecure()` 호출 예제



해당 함수는 HackShield가 정상적으로 초기화된(`_AhnHS_Initialize()` 호출) 후에 호출해야 합니다.

### 지원 패커

현재 LMP 기능에서 지원하는 패커는 다음과 같습니다.

표 19 LMP 지원 패커

| 패커 이름     | 버전         | 비고                                                                     |
|-----------|------------|------------------------------------------------------------------------|
| Themida   | 2.2.0.0    | LMP 지원 가능                                                              |
| Armadillo | V6.4.0.640 | 패킹 옵션에 따라 부분적으로 지원 가능<br>Doc\\Additional\\Packer_HackShield_호환성.pdf 참고 |

# **7장**

## **부가 기능**

### **(데이터 파일/메시지 암호화)**

7.1 데이터 파일/메시지 암호화 기능 /147

7.2 애플리케이션 프로그래밍 /148

7.3 애플리케이션 프로그래밍 인터페이스 /150

## 7.1 데이터 파일/메시지 암호화 기능

---

HackShield의 데이터 파일/메시지 암호화 기능을 소개하고 이 기능을 구현하기 위한 방법을 설명합니다.

### 7.1.1 기능 소개

HsCryptoUtil은 데이터 암/복호화 SDK입니다. 최근 컴퓨터의 발달로 인해 더욱 강력한 암호화 기술이 요구되고 있고 그 결과 DES(Data Encryption Standard)보다 진보된 AES(Advanced Encryption Standard)가 표준으로 지정되어 사용되고 있습니다. HsCryptoUtil은 128bit AES를 사용해서 더욱 강력한 데이터 암/복호화를 제공합니다.

주요 데이터 파일과 서버, 클라이언트 간에 주고 받는 메시지를 암호화합니다. 따라서 데이터 파일이나 메시지가 노출되어도 암호화되어 있으므로 내용을 확인할 수 없어 안전하게 보호할 수 있습니다.

파일 및 메시지 암/복호화 라이브러리로 제공하는 HsCryptLib.lib(Windows용)와 libhscrypt.so(Linux용), HsCryptLib.h를 사용하여 고객은 간단한 방법으로 메시지 및 파일 암/복호화 처리를 할 수 있습니다.

### 7.1.2 기능 특징

HackShield 데이터 파일/메시지 암호화 기능의 특징은 다음과 같습니다.

#### 인터페이스 함수(API) 제공

HsCryptLib에서 제공하는 기능을 편리하게 사용하고, 그 실행 결과 값을 받아 볼 수 있도록 인터페이스 라이브러리를 제공합니다. 제공되는 인터페이스 라이브러리를 사용하여 개발자는 파일 및 메시지를 손쉽게 암/복호화할 수 있습니다.

#### 데이터 암호화 프로그램 제공

파일 암호화 툴인 HsCryptoUtil.exe를 제공하며 암호화된 파일을 HsCryptLib에서 제공하는 API를 사용하여 복호화할 수 있습니다.

### 7.1.3 시스템 구조

HsCryptLib은 독립된 실행 파일 형태가 아닌 SDK를 이용한 라이브러리 형태로 제공됩니다. HsCryptLib의 전체 구조 및 동작 원리는 다음과 같습니다.

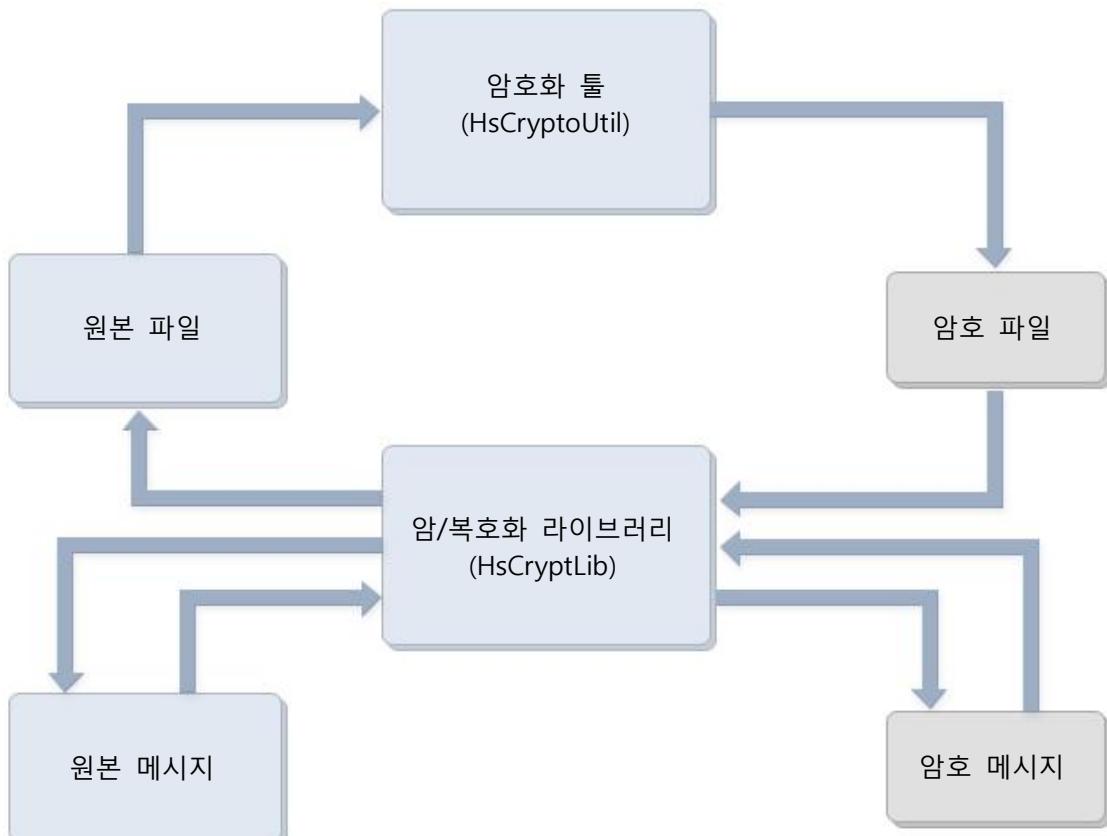


그림 36 HsCryptLib의 전체 구조 및 동작 원리

위 그림의 주요 항목에 대한 설명은 다음과 같습니다.

### 인터페이스 라이브러리

HsCryptLib.lib(Windows용)과 libhscrypt.so(Linux용)가 있습니다. 인터페이스 라이브러리 파일로서 데이터를 암/복호화할 때 사용하는 API를 제공합니다.

### HsCryptoUtil 프로그램

암호화 프로그램으로 HsCryptLib.lib를 이용하여 파일을 암호화하는 기능을 제공합니다.

## 7.2 애플리케이션 프로그래밍

HackShield의 데이터 파일/메시지 암호화 기능을 구현하는 방법을 설명합니다.

### 7.2.1 프로그래밍 순서

게임 클라이언트 개발자가 HsCryptLib 기능을 구현하는 순서는 다음과 같습니다.

1. 함수를 호출하는 코드를 작성하기 전에 제공 받은 HsCryptLib 파일 목록을 확인하고 필요한 파일을 복사합니다.
2. 암/복호화 초기화하는 함수 `_HsCrypt_InitCrypt()` 호출 코드를 작성합니다. 이 함수는 데이터를 암/복호화하기 전에 호출합니다. 초기화하지 않으면 암/복호화를 할 수 없습니다.
3. 암호화된 파일인 경우, 파일을 부분적으로나 전체를 복호화하는 함수 `_HsCrypt_FRead()`이나 `_HsCrypt_GetDecMsg()` 호출 코드를 작성합니다. 이 함수를 호출하여 데이터를 특정 부분이나 전체를 복호화할 수 있으며 복호화된 데이터는 버퍼로 출력됩니다.
4. 메시지를 암호화하는 함수 `_HsCrypt_GetEncMsg()` 호출 코드를 작성하여 메시지를 암호화할 수 있습니다. 암호화된 데이터는 버퍼로 출력됩니다.
5. 메시지를 복호화하는 함수 `_HsCrypt_GetDecMsg()` 호출 코드를 작성하여 메시지를 복호화할 수 있습니다. 복호화된 데이터는 버퍼로 출력됩니다.
6. 작성한 소스 코드가 정상적으로 동작하는지 테스트합니다.
7. 게임 클라이언트를 게임 유저에게 배포합니다.

### 7.2.2 프로그래밍 준비

HsCryptLib를 사용하여 프로그래밍을 시작하기 전에 확인해야 하는 HsCryptLib 관련 파일 목록과 컴파일러 설정 방법은 다음과 같습니다.

#### HsCryptLib 관련 파일

HsCryptLib 관련 파일은 다음과 같습니다.

**표 20 HsCryptLib 파일**

| 파일 이름          | 설치 폴더        | 설명                                      |
|----------------|--------------|-----------------------------------------|
| HsCryptLib.h   | [프로그램 소스 폴더] | 헤더 파일                                   |
| HsCryptLib.lib | [프로그램 소스 폴더] | 윈도우용 라이브러리 파일<br>멀티스레드 및 싱글스레드 라이브러리 제공 |
| libhscrypt.so  | [프로그램 소스 폴더] | 리눅스용 라이브러리 파일                           |

#### 컴파일러 설정

HsCryptLib를 사용하는 암/복호화 프로그램의 프로젝트 파일에는 반드시 HsCryptLib.lib(libhscrypt.so) 파일을 라이브러리나 소스 코드 목록에 포함시켜야 합니다. 단, Windows용 HsCryptLib를 사용하는 프로젝트는 멀티스레드 기반 라이브러리를 사용하는지 싱글스레드 기반 라이브러리를 사용하는지 확인하여 적합한 HsCryptLib.lib를 적용합니다.

## 7.3 애플리케이션 프로그래밍 인터페이스

HackShield의 데이터 파일/메시지 암호화 기능 구현에 필요한 API를 소개합니다.

### 참고

이 장에서 사용하고 있는 샘플 코드는 Microsoft Visual C++ 6.0을 기준으로 한 C/C++ 언어로 작성했습니다. 프로그래밍에 사용되는 언어는 각 프로그램의 특성 및 시스템 환경에 따라서 변경할 수 있습니다.

### 7.3.1 \_HsCrypt\_InitCrypt()

암/복호화를 초기화하는 함수입니다.

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

#### Parameters

| Parameter                                                                                                                                                                                                                              | Description                                                                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| PHSCRYPT_KEYINFO<br>구조체 정의<br>typedef struct_HSCRYPT_KEYINFO<br>{<br>BYTE byInitKey[HSCRYPTLIB_INITKEY_SIZE];<br>BYTE AesEncKey[HSCRYPTLIB_KEY_SIZE];<br>BYTE AesDecKey[HSCRYPTLIB_KEY_SIZE];<br>} HSCRYPT_KEYINFO, *PHSCRYPT_KEYINFO; | [in][out] 암/복호기 구조체<br><br>초기화 키(16bytes)<br>암호 키(550bytes)<br>복호 키(550bytes) |

#### Return Value

**ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 초기화 성공
- 원인:
- 확인사항: N/A

**ERROR\_HSCRYPTLIB\_INITCRYPT\_INVALIDPARAM (Value = 0x0001B002)**

- 설명: 잘못된 파라미터 입력
- 원인: 전달받은 포인터가 null 인 경우
- 확인사항: 전달한 인자가 정상적인지 확인

## WIN32 Defined Error (Value = WIN32 Defined)

### Remarks

\_HsCrypt\_InitCrypt()는 데이터 암/복호화를 하려면 프로그래밍 준비 작업을 완료한 후 반드시 호출해야 합니다. 이 함수에서 암/복호화를 하기 위해 필요한 여러 데이터 값을 설정합니다. HSCRYPT\_KEYINFO 구조체의 byInitKey를 할당한 다음 \_HsCrypt\_InitCrypt() 호출이 성공해야 AesEncKey(암호 키)와 AesDecKey(복호 키)가 생성됩니다. 이 키를 이용하여 메시지 및 파일을 암/복호화합니다.

### Example

\_HsCrypt\_InitCrypt() 호출 예제는 다음과 같습니다.

```
typedef struct _HSCRYPT_KEYINFO
{
    BYTE byInitKey[HSCRYPTLIB_INITKEY_SIZE]; // 초기화키
    BYTE AesEncKey[HSCRYPTLIB_KEY_SIZE]; // 암호키
    BYTE AesDecKey[HSCRYPTLIB_KEY_SIZE]; // 복호키
} HSCRYPT_KEYINFO, *PHSCRYPT_KEYINFO;

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

dwRet = _HsCrypt_InitCrypt ( &HsKeyInfo );
```

그림 37 \_HsCrypt\_InitCrypt() 호출 예제

위 호출 예제에서 HSCRYPT\_KEYINFO 구조체를 선언한 다음 byInitKey에 초기화 키 값을 입력합니다. HsKeyInfo.byInitKey에 들어가는 초기화 키 크기는 반드시 16byte로 설정해야 합니다.

\_HsCrypt\_InitCrypt()를 호출하면 HSCRYPT\_KEYINFO 구조체의 AesEncKey(암호키)와 AesDecKey(복호키)에 실제로 키 값이 할당됩니다. 이 키값을 이용하여 메시지 및 파일을 암/복호화합니다.

---

#### ⚠ 주의

초기화 키에 의해 생성된 암/복호화 키 쌍이 일치해야 파일 및 메시지 암/복호화를 할 수 있습니다.

---

### 7.3.2 \_HsCrypt\_GetEncMsg()

메시지를 암호화한 다음 암호화된 데이터 버퍼로 출력하는 함수입니다.

`_HsCrypt_InitCrypt()`로 암/복호화를 초기화하면 `_HsCrypt_GetEncMsg()`를 호출하여 데이터를 암호화하여 암호화된 데이터를 버퍼로 출력하게 됩니다.

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

## Parameters

| Parameter  | Description   |
|------------|---------------|
| PbyInput   | [in] 암호화할 버퍼  |
| NInLength  | [in] 암호화할 크기  |
| PAesEncKey | [in] 암호화 키    |
| PbyOutput  | [out] 암호화된 버퍼 |

## Return Value

### **ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 메시지 암호화 성공
- 원인: N/A
- 확인사항: N/A

### **ERROR\_HSCRYPTLIB\_GETENCMMSG\_INVALIDPARAM (Value = 0x0001B003)**

- 설명: 잘못된 파라미터 입력
- 원인: 전달된 파라미터가 비정상적인 경우
- 확인사항: 전달한 파라미터 값이 null이나 0 인지 확인

### **WIN32 Defined Error (Value = WIN32 Defined)**

## Example

`_HsCrypt_GetEncMsg()` 호출 예제는 다음과 같습니다.

```
dwRet = _HsCrypt_GetEncMsg (  
    byPlainMsg,           // [in] 암호화할 버퍼  
    sizeof(byPlainMsg),   // [in] 암호화할 크기
```

```

        HsKeyInfo.AesEncKey,    // [in] 암호화키
        byEncMsg                // [out]암호화된 버퍼
    ) ;

```

그림 38 \_HsCrypt\_GetEncMsg() 호출 예제

### 참고

암호화하기 전의 메시지 크기와 암호화된 메시지의 크기는 같습니다.

### 7.3.3 \_HsCrypt\_GetDecMsg()

메시지를 복호화한 다음 복호화된 데이터 버퍼로 출력하는 함수입니다.

```

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

```

#### Parameters

| Parameter  | Description   |
|------------|---------------|
| pbyInput   | [in] 복호화할 버퍼  |
| nInLength  | [in] 복호화할 크기  |
| pAesEncKey | [in] 복호화 키    |
| pbyOutput  | [out] 복호화된 버퍼 |

#### Return Value

**ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 메시지 암호화 성공
- 원인: N/A
- 확인사항: N/A

**ERROR\_HSCRYPTLIB\_GETDECMMSG\_INVALIDPARAM (Value = 0x0001B004)**

- 설명: 잘못된 파라미터 입력
- 원인: 전달된 파라미터가 비정상적인 경우
- 확인사항: 전달한 파라미터 값이 null이나 0인지 확인

**WIN32 Defined Error (Value = WIN32 Defined)**

## Example

\_HsCrypt\_GetDecMsg() 호출 예제는 다음과 같습니다.

```
dwRet = _HsCrypt_GetDecMsg (
    byEncMsg,           // [in] 복호화할 버퍼
    sizeof(byEncMsg),   // [in] 복호화할 크기
    HsKeyInfo.AesDecKey, // [in] 복호화키
    byDecMsg           // [out] 복호화된 버퍼
);
```

그림 39 \_HsCrypt\_GetDecMsg() 호출 예제

### 참고

복호화하기 전의 메시지 크기와 복호화된 메시지의 크기는 같습니다.

### 7.3.4 \_HsCrypt\_FRead()

파일에서 원하는 부분만 복호화하여 데이터를 버퍼로 출력하는 함수입니다.

파일 구조체 포인터를 이용하여 파일을 부분 및 전체 복호화합니다. 암/복호화 키를 초기화하고 fseek()를 호출하여 원하는 파일 포인터로 이동한 다음 데이터를 복호화합니다.

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

### Parameters

| Parameter     | Description          |
|---------------|----------------------|
| lpOutBuffer   | [out] 복호화된 버퍼        |
| dwDecryptSize | [in] 복호화할 크기         |
| pInputStream  | [in] 복호화할 파일 구조체 포인터 |
| pAesDecKey    | [in] 복호 키            |
| pdwReadLen    | [out] 복호화된 크기        |

## Return Value

### **ERROR\_SUCCESS (Value = 0x00000000)**

- 설명: 복호화 성공
- 원인: N/A
- 확인사항: N/A

### **ERROR\_HSCRYPTLIB\_FREAD\_INVALIDPARAM (Value = 0x0001B005)**

- 설명: 잘못된 파라미터 입력
- 원인: 전달된 파라미터가 비정상적인 경우
- 확인사항: 전달된 파라미터 값이 null이나 0인지 확인

### **ERROR\_HSCRYPTLIB\_FREAD\_GETFILELEN (Value = 0x0001B009)**

- 설명: 파일 크기 얻기 실패
- 원인: 파일 핸들이 비정상적인 경우
- 확인사항: 전달된 파일 핸들 값이 정상인지 확인

### **ERROR\_HSCRYPTLIB\_FREAD\_SIZEZERO (Value = 0x0001B00B)**

- 설명: 파일 크기가 0
- 원인: 복호화할 파일이 없는 경우
- 확인사항: 정상적인 파일인지 확인

### **ERROR\_HSCRYPTLIB\_FREAD\_GETPOSITION (Value = 0x0001B00A)**

- 설명: 파일 포인터 현재 위치 얻기 실패
- 원인: 전달 받은 파일 핸들이 비정상적인 경우
- 확인사항: 전달된 파일 핸들 파라미터 확인

### **ERROR\_HSCRYPTLIB\_FREAD\_FSEEK (Value = 0x0001B00C)**

- 설명: 현재 파일 포인터 위치의 블럭으로 이동하는 것에 실패
- 원인: 전달 받은 파일 핸들이 비정상적인 경우
- 확인사항: 전달된 파일 핸들 파라미터 확인

### **ERROR\_HSCRYPTLIB\_FREAD\_DECRYPT\_RANGE (Value = 0x0001B006)**

- 설명: 파일보다 큰 복호화할 블록 크기
- 원인: 복호화할 블록 크기가 파일보다 큰 경우
- 확인사항: 전달한 dwDecryptSize 값이 파일 크기보다 큼지 확인

### **ERROR\_HSCRYPTLIB\_FREAD\_DECRYPT\_FREAD (Value = 0x0001B007)**

- 설명: 파일 읽기 실패

- 원인: 파일 핸들이 비정상적이거나 파일에 락이 걸려 있는 경우
- 확인사항: 파일 핸들이나 다른 프로세스가 해당 파일에 접근하고 있는지 확인

#### **ERROR\_HSCRYPTLIB\_FREAD\_DECRYPT\_GETDECMMSG (Value = 0x0001B008)**

- 설명: 메시지 복호화 실패
- 원인: API 내부 에러
- 확인사항: 안랩 HackShield 기술 지원에 문의

#### **ERROR\_HSCRYPTLIB\_EXCEPTION (Value = 0x0001B001)**

- 설명: 예외 발생
- 원인: N/A
- 확인사항: 안랩 HackShield 기술 지원에 문의

#### **WIN32 Defined Error (Value = WIN32 Defined)**

### **Example**

`_HsCrypt_FRead()` 호출 예제는 다음과 같습니다.

```
dwRet = _HsCrypt_FRead (
    byPlainBuf,           // [out] 복호화된 버퍼
    dwDecSize,            // [in] 복호화할 크기
    InputStream,          // [in] 읽을 파일 포인터
    HsKeyInfo.AesDecKey, // [in] 복호키
    &dwReadLen           // [out] 복호화된 크기
);
```

**그림 40 `_HsCrypt_FRead()` 호출 예제**

위 호출 예제에서 블록 암호(Block Cipher)를 사용하여 암/복호화를 합니다. 복호화를 원하는 부분을 블록으로 나누어서 블록을 복호화하고 필요한 부분만 버퍼에 저장하여 리턴합니다.

파일 전체를 복호화하려면 `fseek()`를 사용하여 파일 구조체 포인터를 파일의 맨 앞으로 설정한 다음 `_HsCrypt_FRead()`의 두 번째 파라미터에 파일 전체 크기를 입력합니다.

---

#### **⚠ 주의**

대상 파일을 암호화했던 암호키의 쌍인 복호키를 입력해야 하며, 키 관리가 어렵다면 암/복호키를 초기화했던 초기화키와 `_HsCrypt_InitCrypt()`를 이용하여 생성된 복호키를 사용합니다.

---

# **8장**

## **부가 기능**

### **(User 권한 실행 지원)**

**8.1 User 권한 실행 지원 기능 /158**

**8.2 애플리케이션 프로그래밍 /159**

**8.3 애플리케이션 프로그래밍 인터페이스 /162**

## 8.1 User 권한 실행 지원 기능

---

HackShield의 User 권한 실행 지원 기능을 소개하고 이 기능을 구현하기 위한 방법을 설명합니다.

### 8.1.1 기능 소개

HsUserUtil은 NT 계열의 운영체제를 사용하는 경우 관리자 계정이 아닌 일반 사용자 계정으로 로그온한 환경에서 HackShield의 게임 해킹 방어 기능이 정상적으로 동작하도록 하는 기능입니다.

제공하는 HsUserUtil.lib를 사용하면 게임 클라이언트 및 HackShield를 관리자 계정으로 로그온하지 않고 일반 사용자 계정으로 로그온하더라도 게임을 실행할 수 있고 HackShield의 게임 해킹 방어 기능이 정상적으로 동작하도록 지원합니다.

### 8.1.2 기능 특징

User 권한 실행 지원 기능의 특징은 다음과 같습니다.

#### 인터페이스 함수(API) 제공

HsUserUtil에서 제공하는 기능을 편리하게 사용하고 그 실행 결과 값을 받아 볼 수 있도록 인터페이스 라이브러리를 제공합니다. 고객은 정책에 따라 게임 클라이언트 및 HackShield를 일반 사용자 계정으로 실행되도록 인터페이스 라이브러리를 사용하여 프로그래밍할 수 있습니다.

#### 테스트 프로그램 제공

HsUserUtil에서 제공하는 API를 사용하여 구현한 테스트용 게임 클라이언트 프로그램인 HsUserUtilTest.exe를 제공합니다. 고객은 테스트 프로그램을 참고하여 HsUserUtil의 기능 및 예제 코드를 확인하고 클라이언트 프로그램 개발에 적용할 수 있습니다.

### 8.1.3 시스템 구조

HsUserUtil은 독립된 실행 파일 형태가 아닌 SDK를 이용한 라이브러리 형태로 제공됩니다. HsUserUtil의 전체 구조 및 동작 원리는 다음과 같습니다.

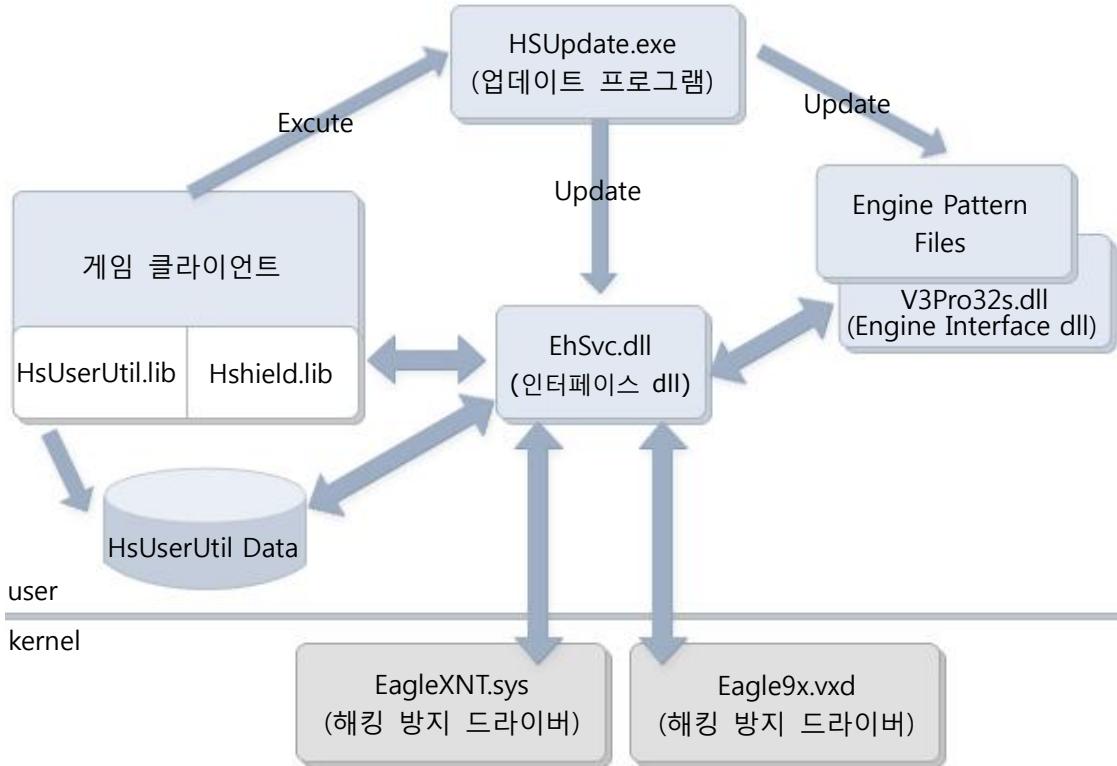


그림 41 HsUserUtil의 전체 구조 및 동작 원리

위 그림의 주요 항목에 대한 설명은 다음과 같습니다.

### HsUserUtil.lib (인터페이스 라이브러리)

인터페이스 파일입니다. 게임이 일반 사용자 계정으로 동작하게 하는 API를 제공합니다.

### HsUserUtil Data

게임이 일반 사용자 계정으로 동작하게 하는 쉐도우 계정 정보 및 관련 데이터를 저장합니다.

### EhSvc.dll (HackShield 모듈)

HackShield 모듈로서 실제 게임에 로드되어 해킹을 방어합니다. 일반 사용자 권한으로 게임이 실행되는 경우 HsUserUtil 데이터의 정보를 이용해서 게임이 정상적으로 실행되고 게임 방어 기능이 정상 동작하도록 지원합니다.

## 8.2 애플리케이션 프로그래밍

HackShield의 User 권한 실행 지원 기능을 구현하는 방법을 설명합니다.

---

### 참고

이 장에서 사용하고 있는 샘플 코드는 Microsoft Visual C++ 6.0을 기준으로 한 C/C++ 언어로 작성했습니다. 프로그래밍에 사용되는 언어는 각 프로그램의 특성 및 시스템 환경에 따라서 변경할 수 있습니다.

---

## 8.2.1 프로그래밍 순서

게임 클라이언트 개발자가 HsUserUtil 기능을 구현하는 순서는 다음과 같습니다.

---

### 참고

이 매뉴얼에서는 게임 클라이언트 프로그램에 HSUserUtil을 사용하는 경우를 예로 설명하고 있습니다. 만약 게임 클라이언트 프로그램 외에 게임 런처(launcher) 프로그램이 별도로 존재하는 경우, HSUserUtil 라이브러리를 게임 런처 프로그램에서 사용할 수도 있습니다. 게임의 구조에 맞게 다르게 적용하십시오.

---

1. 함수를 호출하는 코드를 작성하기 전에 제공 받은 HsUserUtil 파일 목록을 확인하고 필요한 파일을 복사합니다.
2. 쉐도우 계정을 생성하는 함수 `_AhnHsUserUtil_CreateUser()` 호출 코드를 작성합니다. 일반 사용자 계정으로 게임 실행 및 게임 해킹 방어 기능이 동작할 수 있습니다.
3. HackShield 시작 함수 호출 후 폴더의 NTFS 권한 설정 함수 `_AhnHsUserUtil_SetFolderPermission()` 호출 코드를 작성합니다. 일반 사용자 계정에서도 게임에 필요한 파일에 대해 쓰기 기능이 동작할 수 있습니다.
4. 작성한 소스 코드가 정상적으로 동작하는지 테스트합니다.
5. 게임 클라이언트를 게임 유저에게 배포합니다.

지금까지의 과정을 순서대로 간단히 나타내면 다음과 같습니다.

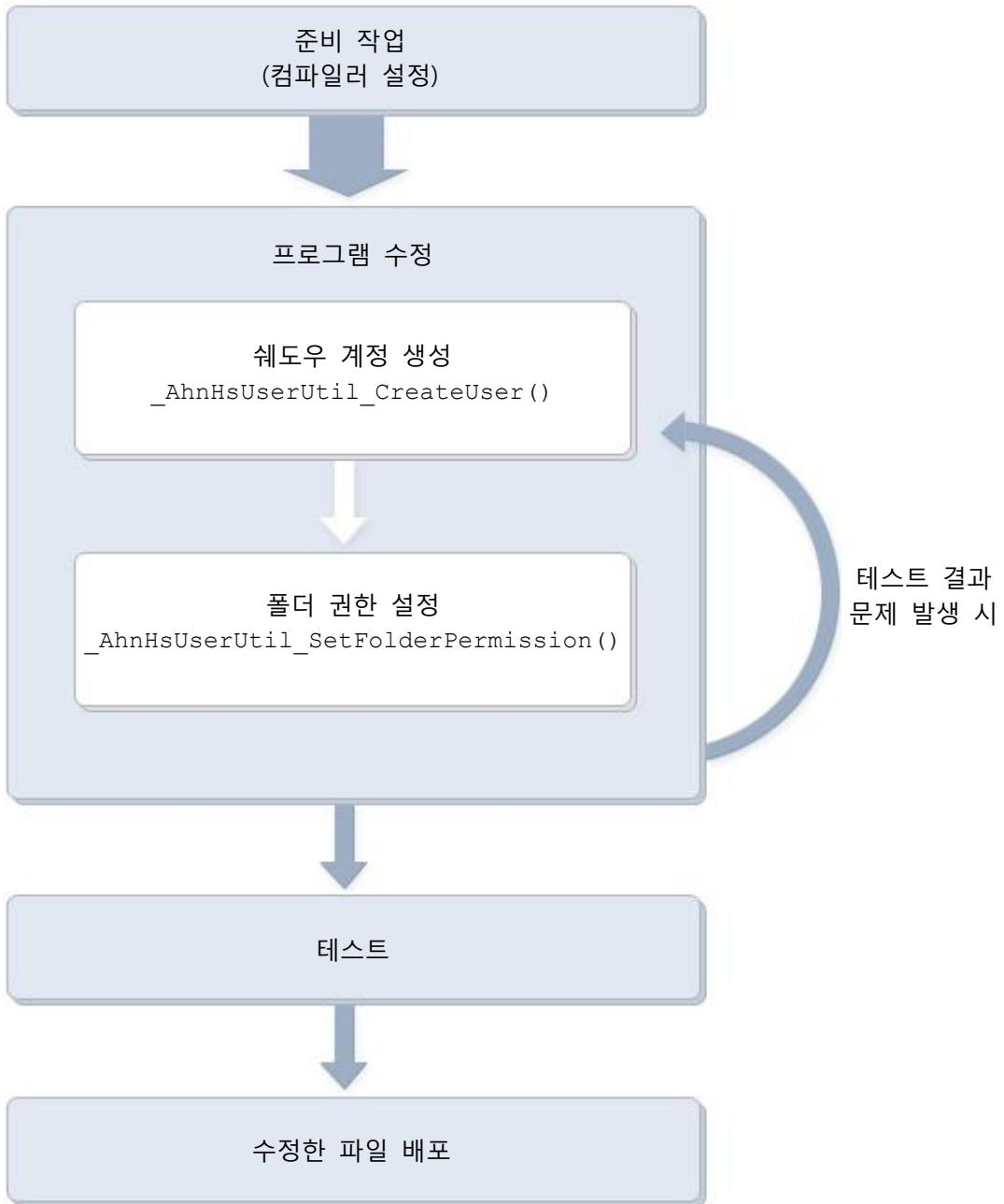


그림 42 HsUserUtil 프로그래밍 순서

### 8.2.2 프로그래밍 준비

HsUserUtil을 사용하여 프로그래밍을 시작하기 전에 확인해야 하는 HsUserUtil 관련 파일 목록과 컴파일러 설정 방법은 다음과 같습니다.

#### HsUserUtil 관련 파일

HsUserUtil 관련 파일은 다음과 같습니다.

표 21 HsUserUtil 파일

| 파일 이름 | 설치 폴더 | 설명 |
|-------|-------|----|
|       |       |    |

|                |            |          |
|----------------|------------|----------|
| HsUserUtil.h   | [게임 소스 폴더] | 헤더 파일    |
| HsUserUtil.lib | [게임 소스 폴더] | 라이브러리 파일 |

## 컴파일러 설정

HsUserUtil을 사용하는 게임 클라이언트 프로그램의 프로젝트 파일에는 반드시 HsUserUtil.lib 파일을 라이브러리나 소스 코드 목록에 포함시켜야 합니다.

## 8.3 애플리케이션 프로그래밍 인터페이스

HackShield의 User 권한 실행 지원 기능 구현에 필요한 API를 소개합니다.

### 8.3.1 \_AhnHsUserUtil\_CreateUser()

일반 사용자 권한으로 로그온했을 때 게임 해킹 방어 기능에 사용할 쉐도우 계정을 생성하는 함수입니다.

```
DWORD __stdcall
_AhnHsUserUtil_CreateUser();
```

#### Parameters

N/A

#### Return Value

##### HSUSERUTIL\_ERR\_OK (Value = 0x00000000)

- 설명: 쉐도우 계정 생성 성공
- 원인: N/A
- 확인사항: N/A

##### HSUSERUTIL\_ERR\_NOT\_ADMIN (Value = 0x0005A002)

- 설명: 관리자 권한 없음
- 원인: 현재 로그온한 계정이 관리자 계정이 아닌 경우
- 확인사항: 적용 방식에 따라 차이가 있겠지만 일반 User 권한에서 \_AhnHsUserUtil\_CreateUser()가 호출될 수 있도록 적용이 된 경우에는 이 에러는 처리하지 않도록 해야함

##### HSUSERUTIL\_ERR\_NOT\_1NT (Value = 0x0005A003)

- 설명: 운영체제 버전이 NT 가 아닌 시스템
- 원인: NT 계열 시스템이 아닌 경우
- 확인사항: 사용 중인 운영체제 버전이 NT 이상인지 확인

#### **HSUSERUTIL\_ERR\_DELSHADOWACNT\_FAIL (Value = 0x0005A005)**

- 설명: 기존에 생성한 쉐도우 계정 삭제 실패
- 원인: 내부 에러
- 확인사항: 안랩 HackShield 기술 지원에 문의

#### **HSUSERUTIL\_ERR\_DELHIDEIDINFO\_FAIL (Value = 0x0005A006)**

- 설명: Windows XP 시작 화면에서 쉐도우 계정을 감추기 위한 정보 삭제 실패
- 원인: 내부 에러
- 확인사항: 안랩 HackShield 기술 지원에 문의

#### **HSUSERUTIL\_ERR\_DELSHADOWACNTINFO\_FAIL (Value = 0x0005A007)**

- 설명: 쉐도우 계정 정보 삭제 실패
- 원인: 내부 에러
- 확인사항: 안랩 HackShield 기술 지원에 문의

#### **HSUSERUTIL\_ERR\_ADDSHADOWACNT\_FAIL (Value = 0x0005A008)**

- 설명: 쉐도우 계정 생성 실패
- 원인: 내부 에러
- 확인사항: 안랩 HackShield 기술 지원에 문의

### **Remarks**

`_AhnHsUserUtil_CreateUser()`는 Administrator 권한을 가진 계정으로 로그온한 상태에서 쉐도우 계정이 생성됩니다. 일반 사용자 권한으로 HackShield의 게임 방어 기능을 정상적으로 사용하려면 최초 한번은 이 함수가 호출돼야 쉐도우 계정이 생성됩니다.

`_AhnHsUserUtil_CreateUser()`는 프로그래밍을 하기 위한 준비 작업이 완료되면 가장 먼저 호출합니다. `_AhnHsUserUtil_CreateUser()` 호출이 성공해야 일반 사용자 계정에서도 게임을 실행하고 게임 해킹을 방어할 수 있습니다.

단, `_AhnHsUserUtil_CreateUser()`는 관리자 계정으로 게임이나 게임 런처 프로그램에서 호출합니다.

### **Example**

`_AhnHsUserUtil_CreateUser()` 호출 예제는 다음과 같습니다.

```
dwRet = _AhnHsUserUtil_CreateUser ();
```

**그림 43 \_AhnHsUserUtil\_CreateUser() 호출 예제**

---

### 참고

\_AhnHsUserUtil\_CreateUser()는 이전에 생성된 쉐도우 계정의 사용자 정보가 없거나 기존에 생성된 쉐도우 계정의 사용자 정보로 로그인되지 않는 경우에만 새로운 사용자를 생성합니다. 또한 쉐도우 계정 명명 규칙에 해당하는 계정을 삭제하는 로직이 포함되어 있어 사용하지 않거나 불필요한 계정은 새로운 쉐도우 계정 생성 시에 자동으로 삭제됩니다.

---

## 8.3.2 \_AhnHsUserUtil\_SetFolderPermission()

게임 클라이언트가 설치된 디렉터리에 일반 사용자 계정에 대한 NTFS 쓰기 권한을 부여해서 일반 사용자 계정으로 로그온해도 게임 클라이언트가 업데이트 및 실행에 필요한 파일 쓰기 작업 등을 하게 하는 함수입니다.

```
DWORD __stdcall  
_AhnHsUserUtil_SetFolderPermission (  
    LPSTR szPath  
) ;
```

### Parameters

| Parameter | Value | Description                                                      |
|-----------|-------|------------------------------------------------------------------|
| szPath    |       | NTFS 권한을 설정할 경로의 전체 경로<br>예: C:\Program Files\My Company\My Game |

### Return Value

#### HSUSERUTIL\_ERR\_OK (Value = 0x00000000)

- 설명: 해당 폴더에 대해 NTFS 권한 부여 성공
- 원인: N/A
- 확인사항: N/A

#### HSUSERUTIL\_ERR\_NOT\_ADMIN (Value = 0x0005A002)

- 설명: 관리자 권한이 아님
- 원인: 현재 로그온한 계정이 관리자 계정이 아닌 경우
- 확인사항: 적용 방식에 따라 차이가 있겠지만 일반 User 권한에서 \_AhnHsUserUtil\_CreateUser()가 호출될 수 있도록 적용이 된 경우에는 이 에러는 처리하지 않도록 해야 함

#### HSUSERUTIL\_ERR\_GETVOLUMEINFO\_FAIL (Value = 0005A111)

- 설명: 드라이버 볼륨 정보 읽기 실패

- 원인: 드라이버 볼륨 정보를 읽는 과정에서 문제가 발생한 경우
- 확인사항: 안랩 HackShield 기술 지원에 문의

## Remarks

게임 클라이언트 프로그램이 설치되는 경로가 사용자에 의해서 임의로 지정이 가능한 경우, 이 함수를 실행할 때 주의해야 합니다. 만약 사용자가 C:나 D: 드라이브 같은 루트 디렉터리에 게임을 설치했다면 이 함수 호출 시 해당 드라이브 전체에 대해 NTFS 권한이 부여됩니다. 이는 보안상 취약점이 될 수 있으므로 주의해서 호출하십시오.

지정된 폴더 하위에 많은 수의 하위 폴더와 파일이 존재할 경우 이 함수가 호출되면 NTFS 권한을 설정하는데 수 초에서 수 분의 시간이 소요될 수도 있습니다. 시간 소요는 NTFS 권한을 부여하는 최초 과정에서만 발생할 수 있으며 이미 권한이 부여된 이후에는 시간에 영향을 주지 않습니다.

게임이 NTFS 볼륨에 설치된 경우에는 해당 폴더에 대해서 일반 사용자 권한으로는 파일 쓰기 권한이 없어 게임이 정상적으로 동작하지 않을 수 있습니다. 예를 들어 게임 모듈에 대한 업데이트 프로그램이 실행되어도 게임이 설치된 폴더에 최신 파일을 쓰지 못하는 상황이 발생한다든지, 게임 실행 중에 게임 데이터를 저장하는데 실패할 수도 있습니다.

일반 사용자 계정에 대하여 게임 설치 폴더에 파일 쓰기 권한을 부여하려면 `_AhnHsUserUtil_SetFolderPermission()`를 사용함으로써 NTFS 쓰기 권한을 부여할 수 있습니다.

## Example

`_AhnHsUserUtil_SetFolderPermission()` 호출 예제는 다음과 같습니다.

```
dwRet = _AhnHsUserUtil_SetFolderPermission( "게임이 설치된 경로" );
```

그림 44 `_AhnHsUserUtil_SetFolderPermission()` 호출 예제

일반 사용자 계정에 대한 NTFS 쓰기 권한을 부여하기 위해서는 해당 경로를 반드시 전체 경로(Full Path)로 전달해야 하며 상대 경로나 올바르지 않은 경로를 전달할 경우에는 오동작할 수 있습니다.

전달된 게임 설치 경로가 NTFS 볼륨이 아니거나 NTFS 파일 시스템을 사용하지 않는 Windows 95, 98, ME 계열의 PC에서는 이 함수가 호출되더라도 아무런 동작을 하지 않습니다.

이 함수가 실행되면 해당 경로에 대해서 Users 그룹에 대한 NTFS 쓰기 권한을 부여하게 되며, 이를 반영하기 위해서는 반드시 admin 권한으로 실행된 상태에서 이 함수가 호출되어야 합니다.

---

### ⚠ 주의

게임 클라이언트 프로그램이 실행하는 경로가 사용자에 따라 다르게 설치되므로 고객은 주의하여 작성합니다. 게임이 C:\나 바탕 화면, 윈도우 폴더 등 어디에 설치될지 모르는 상황에서 이러한

---

폴더에 대한 NTFS 권한을 변경하는 것은 보안상 취약점이 될 수도 있습니다.

---

### 8.3.3 \_AhnHsUserUtil\_DeleteUser()

\_AhnHsUserUtil\_CreateUser()를 통해 생성한 쉐도우 계정을 삭제하는 함수입니다.

```
DWORD __stdcall _AhnHsUserUtil_DeleteUser();
```

#### Return Value

**HSUSERUTIL\_ERR\_OK (Value = 0x00000000)**

- 설명: 쉐도우 계정 삭제 성공
- 원인: N/A
- 확인사항: N/A

**HSUSERUTIL\_ERR\_LOADDLL\_FAIL (Value = 0x0005A004)**

- 설명: 해당 기능 동작에 필요한 DLL 이 로드되지 않음
- 원인: 시스템 폴더에 NETAPI32.DLL이나 ADVAPI32.DLL이 없거나 손상된 경우
- 확인사항: 해당 dll이 정상적으로 존재하는지 확인

**HSUSERUTIL\_ERR\_NOT\_NT (Value = 0x0005A003)**

- 설명: 운영체제 버전이 NT가 아닌 시스템
- 원인: NT 계열 시스템이 아닌 경우
- 확인사항: 사용 중인 운영체제 버전이 NT 이상인지 확인

#### Remarks

\_AhnHsUserUtil\_DeleteUser()는 \_AhnHsUserUtil\_CreateUser()에서 생성한 쉐도우 계정을 삭제합니다.

#### Example

\_AhnHsUserUtil\_DeleteUser() 호출 예제는 다음과 같습니다.

```
dwRet = _AhnHsUserUtil_DeleteUser();
```

**그림 45 \_AhnHsUserUtil\_DeleteUser() 호출 예제**

### 8.3.4 \_AhnHsUserUtil\_IsEnableHSAdminRights()

현재 로그인된 계정이 HackShield가 동작할 수 있는 권한을 가지고 있는지 확인하는 함수입니다.

```
DWORD __stdcall _AhnHsUserUtil_IsEnableHSAdminRights();
```

### Return Value

#### HSUSERUTIL\_ERR\_OK (Value = 0x00000000)

- 설명: 쉐도우 계정 정상 등록
- 원인: N/A
- 확인사항: N/A

#### HSUSERUTIL\_ERR\_NOT\_NT (Value = 0x0005A003)

- 설명: 운영체제 버전이 NT가 아닌 시스템
- 원인: NT 계열 시스템이 아닌 경우
- 확인사항: 사용 중인 운영체제 버전이 NT 이상인지 확인

### Remarks

현재 로그인된 계정이 HackShield가 동작할 수 있는 권한을 가지고 있는지 확인하는 함수 이므로 Admin 권한 이거나, 유저 권한이면서 HackShield 쉐도우 계정이 생성되어 있는 경우에 성공을 리턴하고 그렇지 않은 경우 다른 에러 값을 리턴합니다. 단순히 HackShield 쉐도우 계정이 존재하는지 확인하려면 `_AhnHsUserUtil_CheckHSShadowAccount()`를 사용하십시오.

### Example

`_AhnHsUserUtil_IsEnableHSAdminRights()` 호출 예제는 다음과 같습니다.

```
dwRet = _AhnHsUserUtil_IsEnableHSAdminRights();
```

그림 46 `_AhnHsUserUtil_IsEnableHSAdminRights()` 호출 예제

### 8.3.5 `_AhnHsUserUtil_CheckHSShadowAccount()`

`_AhnHsUserUtil_CreateUser()`를 통해 생성한 쉐도우 계정이 정상적으로 등록됐는지 확인하는 함수입니다.

```
DWORD __stdcall _AhnHsUserUtil_CheckHSShadowAccount();
```

### Return Value

#### HSUSERUTIL\_ERR\_OK (Value = 0x00000000)

- 설명: 쉐도우 계정 정상 등록
- 원인: N/A

- 확인사항: N/A

#### **HSUSERUTIL\_ERR\_NOT\_NT (Value = 0x0005A003)**

- 설명: 운영체제 버전이 NT 가 아닌 시스템
- 원인: NT 계열 시스템이 아닌 경우
- 확인사항: 사용 중인 운영체제 버전이 NT 이상인지 확인

#### **HSUSERUTIL\_ERR\_SHADOWACNT\_NOT\_EXIST (Value = 0x0005A009)**

- 설명: HackShield 쉐도우 계정이 존재하지 않음
- 원인: 현재 시스템에 HackShield 쉐도우 계정이 생성되어 있지 않은 경우
- 확인사항: 해당 에러를 통해서 쉐도우 계정 등록 여부를 확인할 수 있음.  
`_AhnHsUserUtil_CreateUser()`를 통해서 HackShield 쉐도우 계정이 정상적으로 생성됐는지 확인

### **Example**

`_AhnHsUserUtil_CheckHSShadowAccount()` 호출 예제는 다음과 같습니다.

```
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;
}
```

그림 47 `_AhnHsUserUtil_CheckHSShadowAccount()` 호출 예제

### **8.3.6 `_AhnHSUserUtil_IsAdmin()`**

현재 로그인된 계정이 관리자 권한이 있는지 확인하는 함수입니다.

```
BOOL    __stdcall _AhnHSUserUtil_IsAdmin ();
```

## Return Value

### TRUE

- 설명: NT 계열이 아니거나, NT 계열이고 ADMIN 권한 있음
- 원인: N/A
- 확인사항: N/A

### FALSE

- 설명: NT 계열이고, ADMIN 권한 없음
- 원인: N/A
- 확인사항: N/A

## Example

\_AhnHSUserUtil\_IsAdmin() 호출 예제는 다음과 같습니다.

```
if ( TRUE == _AhnHSUserUtil_IsAdmin() )  
{  
    AfxMessageBox ( "TRUE" );  
}  
else  
{  
    AfxMessageBox ( "FALSE" );  
}
```

그림 48 \_AhnHSUserUtil\_IsAdmin() 호출 예제

# 9장

## 툴 사용 방법

9.1 HSBGen 툴 /171

9.2 HSUpSetEnv 툴 /182

오류! 참조 원본을 찾을 수 없습니다. 오류! 참조 원본을 찾을 수 없습니다. /오류!  
책갈피가 정의되어 있지 않습니다.

9.3 SetServerList 툴 /185

9.4 HSBHelper 툴 /187

## 9.1 HSBGen 툴

HSBGen 툴은 서버에 HackShield Briefcase 파일(AntiCpX.hsb)을 생성하여 게임 파일 및 메모리, HackShield의 무결성 여부를 판단합니다. 서버를 다시 시작하지 않고 새로운 버전을 패치하거나 옵션을 설정해서 이전 버전에 대한 연결 컨트롤하는 역할도 합니다.

---

### !

#### 참고

해당 HSBGen 툴은 HackShield 전용이며 4.2 이후 버전입니다.

---

HSBGen 툴로 생성한 HSB 정보 파일은 생성할 때마다 다르므로 게임 서버가 여러 개이면 서버마다 HSB 정보 파일을 다르게 적용시켜 향상된 보안을 유지합니다. 하지만 게임 클라이언트가 동일하다면 메모리 정보 데이터는 모두 동일해야 합니다.

향상된 LMP 기능을 이용하려면 배포한 게임 클라이언트 파일에 필요한 관련 정보를 추가해야 합니다.

### 9.1.1 시스템 요구 사항

HSBGen 툴을 사용하기 위해 필요한 시스템 요구 사항은 다음과 같습니다.

표 22 HSBGen 툴 시스템 요구 사항

| 운영체제                              | 플랫폼      |
|-----------------------------------|----------|
| Windows 2K3, XP(SP1 이상), VISTA, 7 | x86, x64 |

### 9.1.2 툴 사용 방법

HSBGen 툴을 사용하는 순서는 다음과 같습니다.

1. UI 수동 설정 기반의 HSB 정보 파일을 생성합니다.
2. 자동 HSB 정보 파일을 생성합니다.
3. LMP 적용 여부를 확인합니다.

각 순서에 대한 자세한 설명은 다음과 같습니다.

#### UI 수동 설정 기반의 HSB 정보 파일 생성

UI 수동 설정 기반의 HSB 정보 파일을 생성하는 순서는 다음과 같습니다.

---

### ⚠ 주의

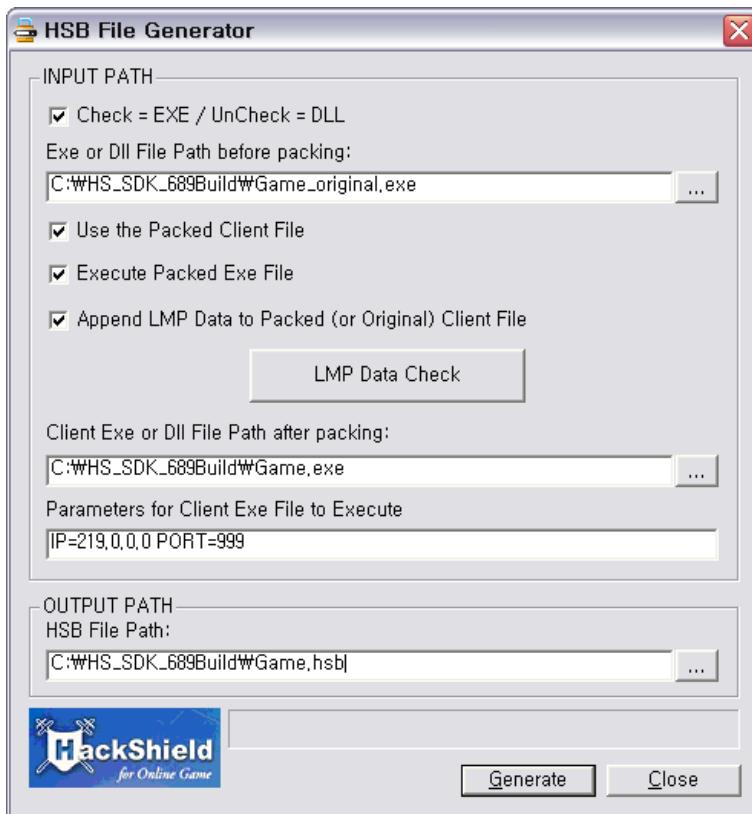
클라이언트 실행 파일은 패킹되기 이전의 원본 릴리즈 파일이어야 합니다.

---

### ⚠ 주의

디지털 서명된 클라이언트 실행 파일을 배포하려면 HSBGen 툴을 사용한 다음 클라이언트 실행 파일에 디지털 서명 작업을 하십시오.

1. \Bin\Win\x86\AntiCrack 의 HSBGen.exe 를 실행합니다.
2. HSB File Generator 창이 아래와 같이 나타나면 INPUT PATH 와 OUPUT PATH 를 설정합니다.



3. 대상 파일이 exe 이면 **Check**, DLL 이면 **UnCheck** 를 합니다.

### ⚠ 주의

대상 파일이 DLL인 경우, 대상 파일 내에 LMP 정보가 자동으로 추가됩니다.

4. 안랩에서 별도로 전달한 HSBGen.ini 파일이 있으면 HSBGen.exe 와 동일한 폴더를 지정합니다. 게임별로 최적화하여 서버 연동의 설정 값을 다르게 적용할 수 있습니다.
5. **Exe or DLL File Path before packing** 입력 상자에 패킹하지 않은 원본 실행 파일의 경로를 설정합니다. …을 눌러 파일을 선택합니다.
6. 패킹된 클라이언트 실행 파일을 배포할 경우, 앞서 설명한 첫 번째 필드에 패킹되기 이전의 파일 경로를 입력합니다.
7. **Use the Packed Client File** 항목을 선택하고 **Client Exe or DLL File Path after packing** 입력 상자에 패킹된 파일 경로를 입력합니다.
8. **Execute Packed Client File** 은 **Client Executable File Path after packing** 에서 입력한 실행 파일을 자동으로 실행한 후 hsb 파일을 생성할 것인지를 결정합니다.

### **⚠ 주의**

Executable Packed Client File 옵션 사용 시, hsb 파일 생성 버튼을 누르면 게임이 진행될 수 있도록 해야 합니다. 따라서 파라미터 정보나 게임 실행 시 필요한 파일을 준비하여 실제로 게임할 수 있는 환경을 준비하십시오.

**9. Parameters for Client Executable File to Execute** 항목에는 **Execute Packed Client File**에서 입력한 실행 파일을 실행 시 필요한 값이 있는 경우 해당 값을 입력합니다.

**10. Append LMP Data to Packed (or Original) Client File** 항목은 LMP 기능을 이용하고자 할 경우, 배포되는 게임 클라이언트 파일에 LMP 정보를 추가합니다. 패킹되지 않은 원본 실행 파일이나 패킹된 클라이언트 실행 파일 모두 LMP 정보로 입력할 수 있습니다.

### **⚠ 주의**

패킹 프로그램이 upx 이외의 다른 패커(themida, asprotect 등..)로 패킹된 경우 execute client file 항목을 선택하십시오.

**11. HSB File Path** 입력 상자에 .hsb 파일을 생성할 전체 경로를 설정합니다. …을 누르면 파일 생성창이 열립니다. 단, 실행 파일과 동일한 경로를 설정하면 클라이언트 파일과 같이 배포될 수 있다는 경고창이 나타납니다. 이것은 사용자에게 환기시키기 위한 목적으로 나타나는 정상 메시지입니다.

**12.** 모든 설정이 완료되면 **Generate** 버튼을 누릅니다.

**13.** 진행 상황이 프로그레스 막대에 표시되며, 완료됐다는 메시지가 나타납니다.

**14.** hsb 파일이 생성되면 해당 hsb 파일을 서버에 패치합니다.

**15.** hsb 파일 생성 시 사용된 게임 클라이언트 파일을 패치합니다.

## **자동 HSB 정보 파일 생성**

HSBGen은, HSB 파일 생성에 필요한 인자정보를 지정함으로써, HSB 파일을 자동으로 생성하는 기능을 제공합니다. 사용 방법은 다음과 같습니다.

```
HSBGen.exe  
[원본게임경로] [패킹여부] [패킹게임경로] [파일실행여부] [파라미터] [HSB경로] [파일타입] [LMP]
```

각 항목에 대한 자세한 설명은 다음과 같습니다.

- 원본게임경로: HSB 정보를 추출할 게임의 경로
- 패킹여부: 패킹된 실행 파일 지원여부(1 = 지원/0 = 지원 안 함)
- 패킹게임경로: HSB 정보를 추출할 패킹된 게임의 경로 (패킹여부가 1 일 경우만 사용)
- 파일실행여부
  - 패킹된 실행 파일 실행 여부(1 = 실행/0 = 실행 안 함)
  - 코드 변조 방지 기능이 있는 패커(Themida 등)로 패킹된 경우 실행

- 파라미터
  - 패킹된 파일 실행 파라미터
  - 파일실행여부가 1인 경우에만 사용
  - 파일실행여부가 0인 경우 생략
- HSB 경로: HSB 파일 경로(절대 경로)
- 파일타입: 보호대상 파일 TYPE(1= EXE, 0 = DLL)
- LMP: LMP 정보 추가 여부(1 = 추가, 0 = 추가 X)

### **⚠ 주의**

인자정보는 '' 단위로 구분하며 구분자 앞 뒤로 공백을 허용하지 않습니다.  
경로명 내에 공백이 있는 경우라도 ""의 입력은 허용하지 않습니다.

HSBGen.exe 사용 예제는 다음과 같습니다.

```
[원본게임경로] C:\HS_SDK\Amazon_ori.exe
[패킹게임경로] C:\HS_SDK\Amazon.exe
[HSB파일경로] C:\HS_SDK\AntiCrack\AntiCrack.hsb
[파라미터] aaa bbb
```

| 예제번호    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 |
|---------|---|---|---|---|---|---|---|---|
| 패킹여부    | O | O | O | O | X | X | O | X |
| 파일실행여부  | O | O | X | X | X | X | X | X |
| 파일타입    | O | O | O | O | O | O | X | X |
| LMP적용여부 | O | X | O | X | O | X | O | O |

### 사용 예제 1

```
HSBGen.exe [게임경로] [패킹지원여부:1] [패킹게임경로] [패킹파일실행:1]
[파라미터] [HSB 저장 경로] [파일 타입:1] [LMP 적용 여부:1]
```

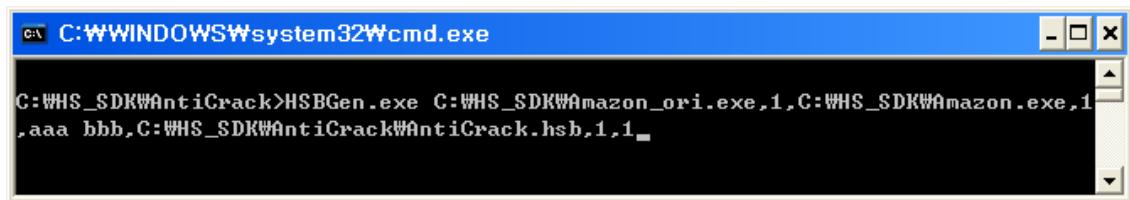
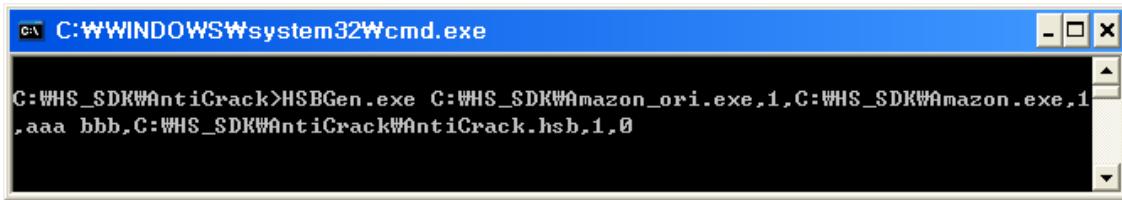


그림 49 Command-line 방식의 HSBGen.exe ([패킹:1], [실행:1], [EXE:1], [LMP:1])

### 사용 예제 2

```
HSBGen.exe [게임경로] [패킹지원여부:1] [패킹게임경로] [패킹파일실행:1]
[파라미터] [HSB 저장 경로] [파일 타입:1] [LMP 적용 여부:0]
```

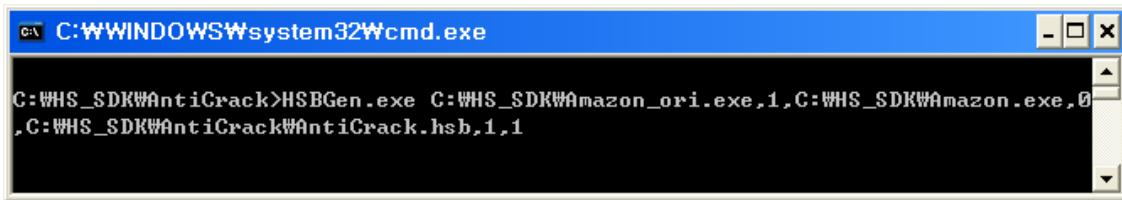


```
C:\WHS_SDK\AntiCrack>HSBGen.exe C:\WHS_SDK\Amazon_ori.exe,1,C:\WHS_SDK\Amazon.exe,1,aaa bbb,C:\WHS_SDK\AntiCrack\AntiCrack.hsb,1,0
```

그림 50 Command-line 방식의 HSBGen.exe ([패킹:1], [실행:1], [EXE:1], [LMP:0])

#### 사용 예제 3

HSBGen.exe [게임경로] [패킹지원여부:1] [패킹게임경로] [패킹파일실행:0]  
[파라미터] [HSB 저장 경로] [파일 탑입:1] [LMP 적용 여부:1]

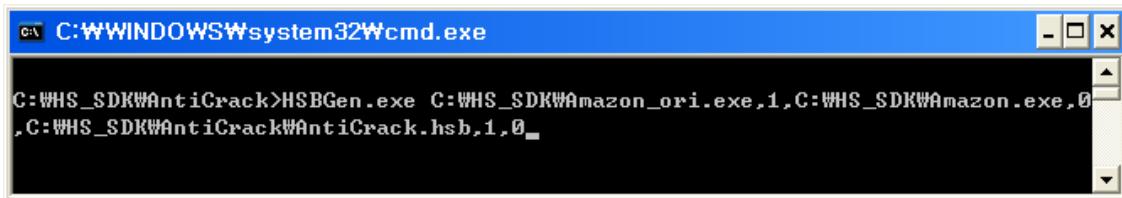


```
C:\WHS_SDK\AntiCrack>HSBGen.exe C:\WHS_SDK\Amazon_ori.exe,1,C:\WHS_SDK\Amazon.exe,0,C:\WHS_SDK\AntiCrack\AntiCrack.hsb,1,1
```

그림 51 Command-line 방식의 HSBGen.exe ([패킹:1], [실행:0], [EXE:1], [LMP:1])

#### 사용 예제 4

HSBGen.exe [게임경로] [패킹지원여부:1] [패킹게임경로] [패킹파일실행:0]  
[파라미터] [HSB 저장 경로] [파일 탑입:1] [LMP 적용 여부:0]

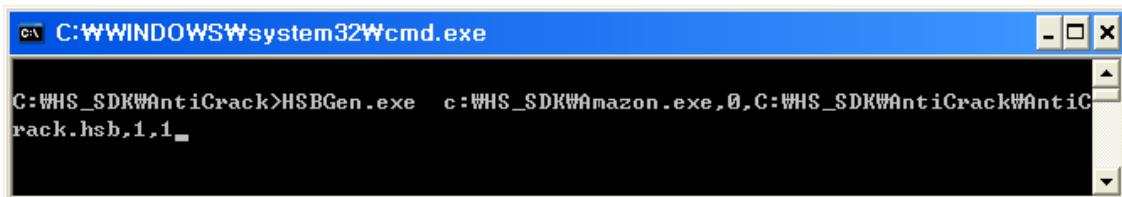


```
C:\WHS_SDK\AntiCrack>HSBGen.exe C:\WHS_SDK\Amazon_ori.exe,1,C:\WHS_SDK\Amazon.exe,0,C:\WHS_SDK\AntiCrack\AntiCrack.hsb,1,0
```

그림 52 Command-line 방식의 HSBGen.exe ([패킹:1], [실행:0], [EXE:1], [LMP:0])

#### 사용 예제 5

HSBGen.exe [게임경로] [패킹지원여부:0] [HSB 저장 경로] [파일 탑입:1]  
[LMP 적용 여부:1]

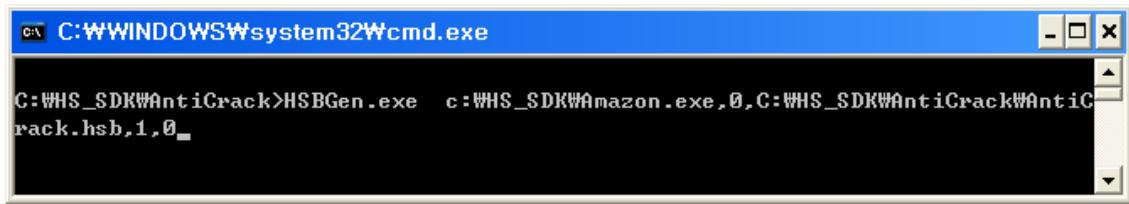


```
C:\WHS_SDK\AntiCrack>HSBGen.exe c:\WHS_SDK\Amazon.exe,0,C:\WHS_SDK\AntiCrack\AntiCrack.hsb,1,1
```

그림 53 Command-line 방식의 HSBGen.exe ([패킹:0], [실행:0], [EXE:1], [LMP:1])

#### 사용 예제 6

HSBGen.exe [게임경로] [패킹지원여부:0] [HSB 저장 경로] [파일 탑입:1]  
[LMP 적용 여부:0]

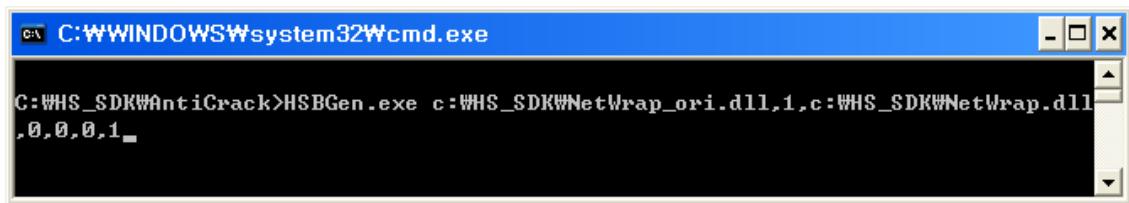


```
C:\WINDOWS\system32\cmd.exe
C:\WHS_SDK\AntiCrack>HSBGen.exe c:\WHS_SDK\Amazon.exe,0,c:\WHS_SDK\AntiCrack\AntiCrack.hsb,1,0
```

그림 54 Command-line 방식의 HSBGen.exe ([패킹:0], [실행:0], [EXE:1], [LMP:0])

#### 사용 예제 7

HSBGen.exe [게임경로] [패킹지원여부:1] [패킹게임경로] [패킹파일실행:0]  
[HSB 저장 경로:0] [파일 탑입:0] [LMP 적용 여부:1]

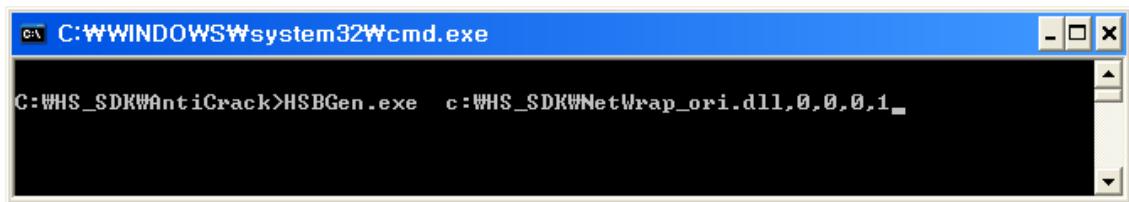


```
C:\WINDOWS\system32\cmd.exe
C:\WHS_SDK\AntiCrack>HSBGen.exe c:\WHS_SDK\NetWrap_ori.dll,1,c:\WHS_SDK\NetWrap.dll,0,0,0,1
```

그림 55 Command-line 방식의 HSBGen.exe ([패킹:1], [실행:0], [EXE:0], [LMP:1])

#### 사용 예제 8

HSBGen.exe [게임경로] [패킹지원여부:0] [HSB 저장 경로:0] [파일 탑입:0]  
[LMP 적용 여부:1]



```
C:\WINDOWS\system32\cmd.exe
C:\WHS_SDK\AntiCrack>HSBGen.exe c:\WHS_SDK\NetWrap_ori.dll,0,0,0,1
```

그림 56 Command-line 방식의 HSBGen.exe ([패킹:0], [실행:0], [EXE:0], [LMP:1])

### RETURN VALUE

#### ERROR\_HSBGEN\_SUCCESS (Value = 0)

- 설명: HSBGEN 작업 성공
- 확인사항: N/A

#### ERROR\_HSBGEN\_INVALID\_PARAMETER (Value = 20001)

- 설명: 잘못된 입력 값
- 확인사항: 입력한 파라미터가 올바른지 확인

**ERROR\_HSBGEN\_CLIENTFILE\_FILE\_PATH (Value = 20002)**

- 설명: 클라이언트 파일 열기 실패
- 확인사항: 입력한 클라이언트 파일 경로가 올바른지 확인

**ERROR\_HSBGEN\_PACKED\_CLIENTFILE\_FILE\_PATH (Value = 20003)**

- 설명: 패킹된 클라이언트 파일 열기 실패
- 확인사항: 입력한 패킹된 클라이언트 파일 경로가 올바른지 확인

**ERROR\_HSBGEN\_OUTPUT\_FOLDER\_PATH (Value = 20005)**

- 설명: HSB 저장 경로가 존재하지 않거나 접근 권한 없음
- 확인사항: 입력한 HSB 파일 경로가 올바른지 확인

**ERROR\_HSBGEN\_INIFILE\_FILE\_PATH (Value = 20006)**

- 설명: HSBGen.ini 파일 열기 실패
- 확인사항: HSBGen.ini 파일의 속성 및 경로 확인

**ERROR\_HSBGEN\_CLIENTFILE\_BAD\_FORMAT (Value = 20007)**

- 설명: 클라이언트 파일이 잘못된 포맷
- 확인사항: 클라이언트 파일이 실행될 수 있는 상태인지 확인

**ERROR\_HSBGEN\_CREATE\_HSB\_FILE\_FAILED (Value = 20008)**

- 설명: HSB 파일 생성 실패
- 확인사항: HSB 파일의 속성 및 경로 확인

**ERROR\_HSBGEN\_NOT\_ENOUGH\_MEMORY (Value = 20009)**

- 설명: 메모리가 부족하여 메모리 할당 실패
- 확인사항: 클라이언트의 메모리 사용량 확인

**ERROR\_HSBGEN\_INVALID\_ADDRESS (Value = 20010)**

- 설명: 잘못된 메모리 영역에 접근
- 확인사항: HSBGen.ini 파일의 속성 및 경로 확인

**ERROR\_HSBGEN\_INIFILE\_WRITE\_FAILED (Value = 20011)**

- 설명: HSBGen.ini 파일에 쓰기 실패
- 확인사항: HSBGen.ini 파일이 읽기 전용 속성인지 확인

**ERROR\_HSBGEN\_ALREADY\_INJECTED (Value = 20012)**

- 설명: HSBGen.dll 파일이 이미 인젝션
- 확인사항: HSBGen.dll 이 이미 인젝션 됐는지 확인

#### **ERROR\_HSBGEN\_MODULE\_FORMAT\_ERROR (Value = 20013)**

- 설명: 잘못된 대상 파일의 DLL/EXE 정보
- 확인사항: 클라이언트가 정상적인 PE 포맷 파일인지 확인

#### **ERROR\_HSBGEN\_INIFILE\_WRITE\_FAILED (Value = 20015)**

- 설명: 잘못된 LMPINFO 정보
- 확인사항: 입력한 파라미터 중 [LMP 정보추가여부]가 0 또는 1 인지 확인

#### **ERROR\_HSBGEN\_RUNNING\_EXEINFO\_ERROR (Value = 20016)**

- 설명: 잘못된 RUNNING\_EXEINFO 정보
- 확인사항: 입력한 파라미터 중 [파일실행여부]가 0 또는 1 인지 확인

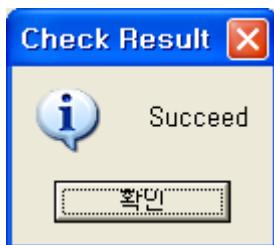
#### **ERROR\_HSBGEN\_INTERNAL\_ERROR (Value = 20017)**

- 설명: 알 수 없는 오류
- 확인사항: 안랩 HackShield 기술 지원에 문의

### **LMP 적용 여부 확인**

LMP 적용 여부를 확인하는 순서는 다음과 같습니다.

1. **Exe or DLL File Path before packing** 입력 상자에 확인할 파일 경로를 설정합니다. ... 을 눌러 파일을 선택합니다.
2. 파일 경로를 설정했으면 **LMP Data Check** 를 누릅니다.
3. 경로가 설정된 파일에 LMP 가 적용돼 있으면 다음과 같은 메시지 박스가 나타납니다.



4. 경로가 설정된 파일에 LMP 가 적용돼 있지 않으면 다음과 같은 메시지 박스가 나타납니다.



## HSBGen.ini

HSBGen.ini 내용은 다음과 같습니다.

```
[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
```

그림 57 HSBGen.ini

HSBGen.ini에 대한 자세한 설명은 다음과 같습니다.

### ■ VERCT

- GrantOldSession: 위 값은 1이나 0 값을 가질 수 있습니다.
- '1'이라면 현재 hsb 파일에 맞는 클라이언트 버전 이외에도 전에 올렸던 hsb 파일에 대한 클라이언트 버전도 허용하겠다는 의미입니다.
- MaxAllowedNumber: GrantOldSession이 1 일때만 의미가 있는 값이고 항상 1 이상의 값을 가져야 합니다. 예를 들어 '1'이라고 하면 현재 적용된 hsb에 대한 클라이언트에만

허용하겠다는 의미이고, '5'라면 이전에 적용된 hsb 파일에 대한 클라이언트 버전을 4 개까지 허용하겠다는 의미입니다.

#### ■ REQRE

- 서버 연동을 할 때 검사하는 기능은 GUID 검사, 클라이언트 파일 검사, HackShield 모듈 검사, 메모리 검사, HackShield 동작 상태 검사 이렇게 총 5 가지가 있습니다.
- GUID 검사: 1
- 클라이언트 파일 검사: 2
- 메모리 검사: 4
- HackShield 엔진검사: 8
- HackShield 동작 상태 검사: 16
- HackShield 동작 상태 강화 검사: 64

---

#### ! 참고

'HackShield 동작 상태 검사'기능은 명시하지 않더라도, Default로 모든 Step(InitStep1, InitStep2, InitStep3, InitStep4)에 적용됩니다.

---

- 각 검사 옵션은 1, 2, 4, 8, 16, 64의 십진수로 이루어져 있으며 중복해서 사용이 가능합니다.
  - InitStep 1: GUID(1) 만 검사하거나 GUID(1) 검사와 HackShield 동작 상태 검사(16)의 동시 검사(17) 만 가능합니다.
  - InitStep 2: GUID 검사(1) 및 클라이언트 파일 검사(2)를 사용할 수 있습니다.
  - InitStep 3: 모든 검사를 사용할 수 있습니다. 예를 들어 '31' 값을 준다면 모든 검사를 사용하겠다는 의미이고 '5'를 준다면 메모리 검사와 HackShield 모듈 검사를 하겠다는 의미입니다.
  - InitStep 4: 위 3 개의 스텝이 완료가 되고 주기적으로 반복하게될 검사를 지정합니다. 보통 메모리 검사 및 HackShield 동작 상태 검사 옵션인 '20'을 사용하십시오.

#### ■ DELAY

- UseDelayedSpiking: '1'을 사용하여 기능을 사용할 경우 \_AhnHS\_VerifyResponse()에서 에러를 리턴할 때 바로 에러를 발생하지 않고 일정 딜레이를 발생시켜 해커들의 혼란을 야기합니다. '0'일 경우 위 기능을 사용하지 않습니다.
- MinDelayCount, MaxDelayCount: 딜레이 스파이킹 기능이 적용될 경우 MinDelayCount 와 MaxDelayCount 사이에 랜덤한 값으로 딜레이가 적용됩니다. 1 은 함수가 호출되는 하나의 주기를 의미합니다.

#### ■ CKMEM

- PageGroupSize, QueryPages: 메모리 검사할 때 한번에 검사되는 메모리 페이지 수를 지정하는 옵션입니다.
- 기본 값 사용을 권장합니다.

#### ■ FPATH

- HSBGen.exe 툴을 통하여 사용자가 입력한 정보로, 게임 클라이언트의 메모리 CRC 정보 파일인 .hsb 파일 생성을 위한 게임 클라이언트 설정 정보입니다.
- ClientFileName: 패킹되지 않은 게임 클라이언트가 위치한 경로입니다.
- UsePackedClientFile: 패킹한 게임 클라이언트를 사용할지 여부에 대한 설정입니다. 패킹한 게임 클라이언트를 사용한다면 이 값은 1이며, 패킹한 게임 클라이언트를 사용하지 않는다면 이 값은 0입니다.
- PackedClientFileName: 패킹된 게임 클라이언트가 위치한 경로입니다. UsePackedClientFile 값이 1로 설정된 경우엔 반드시 필요한 정보입니다.
- GetInfoFromRunningEXE: 패킹된 게임 클라이언트를 사용하는 경우, 게임 클라이언트를 실행시켜 메모리 CRC 정보를 추출하기 위한 설정 값입니다. 이 값이 1이면 PackedClientFileName에 설정된 경로의 파일을 실행시켜 메모리 CRC 정보를 추출하게 됩니다.
- Parameters: 게임 클라이언트 실행 시 필요한 파라미터 정보입니다. PackedClientFileName에 경로가 설정되고, GetInfoFromRunningEXE 값이 1인 경우, 이 값이 유효합니다.
- AppendLMPInfoToClientFile: LMP 기능을 이용하기 위하여 필요한 관련 정보를 게임 클라이언트 파일에 추가하기 위한 설정 값입니다. 해당 값이 '1'이면 LMP 기능을 이용하기 위하여 필요한 관련 정보를 게임 클라이언트 파일에 추가하고, 해당 값이 '0'이면 작업을 진행하지 않습니다.
- HsbFileName: 게임 클라이언트의 메모리 CRC 정보 파일(.hsb 파일)이 생성될 경로입니다. 이 값을 설정하지 않으면 .hsb 파일을 생성할 수 없습니다. 반드시 .hsb 확장자로 설정해야 합니다.
- UseEXE: 대상 파일이 EXE일 경우는 1, DLL인 경우는 0으로 설정합니다.

---

### ⚠ 주의

SDK에 있는 기본 hsbgene.ini를 열면 FPATH 경로가 없습니다. 최초 hsb 파일을 만들 때부터 해당 FPATH 정보가 저장됩니다.

---

#### ■ CKHSB

- UseHSB: 신규 게임 클라이언트 파일 배포시, 해당 게임 클라이언트 파일을 대상으로 하여 HSBGen.exe 툴을 실행하여 생성된 HSB 파일을 게임 서버에 업로드할지의 여부에 대한 설정 값입니다. 값이 '0' (UseHSB=0) 이면 '신규 게임 클라이언트 파일을 대상으로 생성된 HSB 파일을 게임 서버에 업로드하지 않고서 서버 연동하는 것'을 의미하며, 해당 값이 '1' (UseHSB=1) 이면 '신규 게임 클라이언트 파일을 대상으로 생성된 HSB 파일을 게임 서버에 업로드하고서 서버 연동하는 것'을 의미합니다.

---

### ⚠ 주의

[CKHSB] 섹션의 UseHSB=0인 경우, 기능 사용 시 위의 섹션 내용이 적용된 HSB 파일을 최초 한번 서버에 반드시 업로드해야 합니다.

---

---

신규 게임 클라이언트 파일 배포 전에, HSBGen 툴을 이용하여 정상적으로 HSB 파일이 생성이 되었는지 확인하십시오. 단, 해당 HSB 파일을 게임 서버에 업로드할 필요는 없습니다. 해당 HSB 파일 생성 시 HSBGen.ini 내용이 이후에 접속되는 게임 클라이언트에 계속 반영됩니다.

클라이언트 파일 검사와 메모리 검사는 지원하지 않습니다. [REQRE] 섹션 내의 InitStep2, InitStep3, InitStep4에 클라이언트 파일 검사 값 및 메모리 검사 값을 의미하는 숫자 2와 4를 빼고 적용하십시오.

예제:   InitStep1=1  
          InitStep2=1  
          InitStep3=8  
          InitStep4=1

---

### ⚠ 주의

HSBGen.exe 툴을 이용하여 새롭게 생성된 HSB 파일의 내용이 적용되는 시점은 해당 HSB 파일 생성 시에 입력했던 게임 클라이언트 파일이 서버에 접속한 이후입니다.

즉, 새롭게 생성된 HSB 파일을 게임 서버에 업로드했다고 해서 해당 내역이 반영되는 것이 아니라 HSB 파일 생성 시에 입력했던 해당 게임 클라이언트가 최초 한 번 게임 서버에 접속해야 적용됩니다.

---

## 9.2 HSUpSetEnv 툴

---

HSUpSetEnv 툴은 업데이트 환경 설정 파일 (HSUpdate.env)을 생성합니다. FTP/ HTTP 방식을 선택할 수 있으며 다중 URL을 지원합니다.

### ➊ 참고

해당 HSBGen 툴은 HackShield 전용이며, 5.1 이후 버전입니다.

---

### 9.2.1 툴 사용 방법

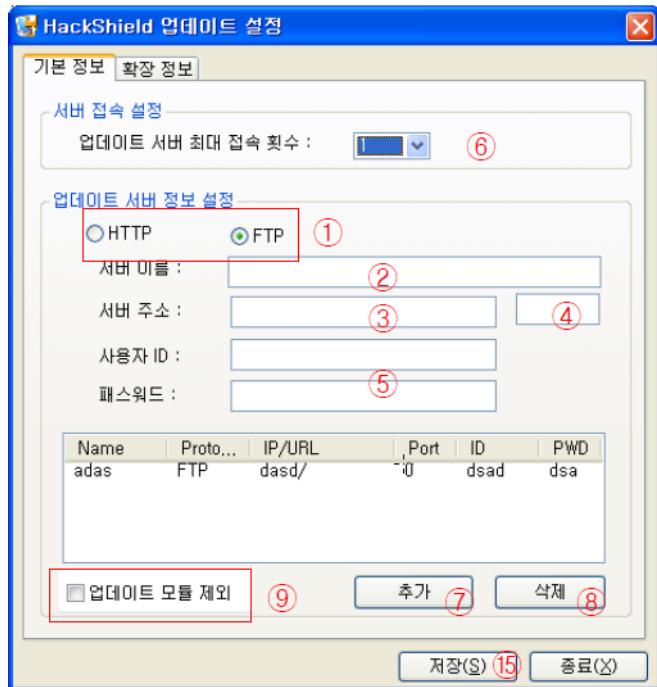
HSUpSetEnv 툴을 사용하여 HSUpdate.env를 생성하는 순서는 다음과 같습니다.

### ⚠ 주의

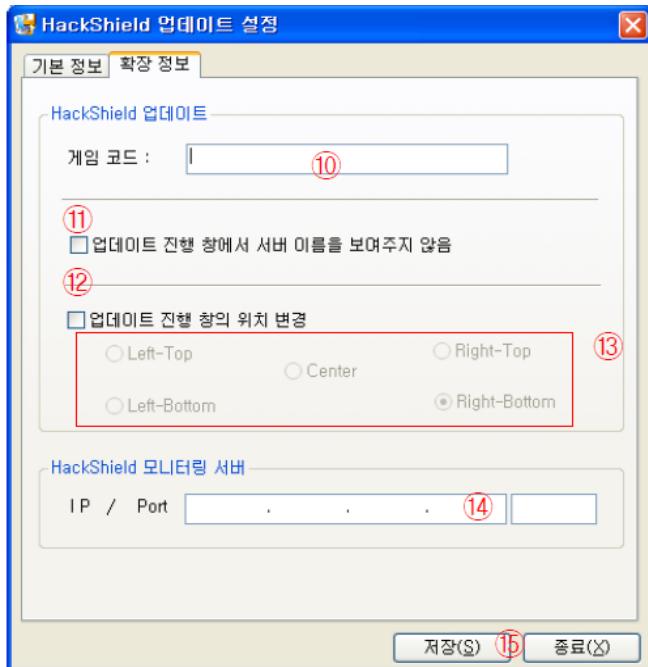
HSUpSetEnv.exe 는 업데이트 설정 파일을 생성하는 툴이므로 배포하지 마십시오.

---

1. [HackShield SDK]\Bin\Win\x86\Util\HSUpSetEnv.exe 를 실행합니다
2. **기본 정보** 탭과 **확장 정보** 탭으로 구성되어 있는 HSUpSetEnv 툴이 나타납니다.
3. **기본 정보** 탭의 각 항목을 설정합니다.



- ① 업데이트 서버에서 사용할 프로토콜을 선택합니다. HTTP를 권장합니다.
  - ② 서버 이름을 입력합니다.
  - ③ 서버 주소를 입력합니다. PatchSet을 저장한 폴더의 경로까지를 지정합니다.
  - ④ 설정해놓은 포트 번호를 입력합니다.
  - ⑤ FTP의 경우, 계정 정보를 설정합니다. 입력하지 않으면 anonymous로 접속합니다.
  - ⑥ 업데이트 실패 시 재시도 횟수를 설정합니다. 전체 리스트 접속이 실패하면 설정한 재접속 횟수만큼 시도합니다.
  - ⑦ 상단의 서버 리스트에 서버를 추가합니다. 업데이트 서버를 둘 이상 운영하는 경우 계속 추가합니다.
  - ⑧ 상단의 서버 리스트에서 선택한 서버 정보를 삭제합니다.
  - ⑨ HackShield 업데이트 관련 파일들을 패치 대상에서 제외합니다. HackShield 업데이트 관련 파일인 ahnupctl.dll, ahnupgs.dll, hsinst.dll, hsupdate.exe, v3hunt.dll, v3inetgs.dll의 변경 내용이 업데이트 서버에 업로드되더라도 해당 파일이 게임 유저의 시스템에 반영하지 않습니다.
- 4. 기본 정보** 탭의 각 항목을 설정합니다.



⑩ 게임 코드의 입력을 권장합니다. `_AhnHS_HSUpdate()`가 아닌 `_AhnHS_HSUpdateEx()`를 사용한 경우에만 유효한 정보입니다.

⑪ 업데이트 진행 알림창의 업데이트 이미지인 HSUPDATE.jpg에서 우측 하단의 업데이트 서버 이름을 숨깁니다.

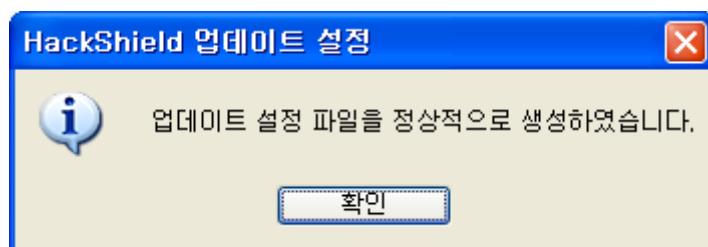
⑫ 업데이트 진행 알림창의 업데이트 이미지인 HSUPDATE.jpg가 삽입된 위치를 변경합니다. 위치 변경을 하지 않으면 우측 하단에 삽입됩니다.

⑬ 업데이트 진행 알림창의 업데이트 이미지인 HSUPDATE.jpg 삽입 위치를 변경하는 경우, 원하는 위치를 지정합니다.

⑭ HackShield 모니터링 기능 사용 시 모니터링 IP 주소와 포트 정보를 입력합니다. 모니터링 기능을 사용하려면 `_AhnHS_StartMonitor()`를 호출하십시오. 단, 모니터링 기능을 사용하지 않을 경우에는 해당 정보를 비워두어야 합니다.

5. 설정이 끝났으면 **저장**을 누릅니다.

6. HSUpdate.env 파일이 정상적으로 생성되는 경우, 다음과 같은 메시지가 나타납니다.



7. 모든 설정이 끝나면 툴을 종료하기 위해 **종료**를 누릅니다.

## 9.3 SetServerList 툴

SetServerList 툴은 afs.dat를 생성합니다.

### 참고

해당 SetServerList 툴은 HackShield 전용이며, 5.1 이후 버전입니다.

ANTIFREESERVER 기능은 유효 IP주소 범위(White-list) 지정하는 방식과 차단 IP주소 범위(Black-list) 지정하는 방식이 있습니다. 두 가지 방식 중 한 가지 방식을 선택해야 합니다. 선택한 방식은 모든 리스트 내의 항목에 반영됩니다.

두가지 방식 모두 클라이언트에서 AHNHS\_CHKOPT\_ANTIFREESERVER 옵션과 사용하십시오. 두 가지 방식에 대한 자세한 설명은 다음과 같습니다.

#### ■ 유효 IP 주소 범위(White-list) 지정 방식

- 감지 콜백이 발생하는 경우
  - 게임 프로세스가 유효 IP 주소 범위 외의 주소로 접속하려는 경우
  - 로컬(127.0.0.1)로 접속하려는 경우
- 게임이 게임 서버에 접근하려는 것을 해킹하여 Bot 서버로 접근하는 유형의 해킹툴 대응에 유용한 방식

#### ■ 차단 IP 주소 범위(Black-list) 지정 방식

- 실행된 모든 프로세스를 대상으로 외부와 연결된 접속 세션이 차단 IP 주소 범위 내에 존재하는지를 확인하는 방식
- 서버와 통신을 하는 해킹툴의 대응에 유용
- 클라이언트 게임 콜백은 발생하지 않으며 서버 연동에 의해 종료되도록 \_AhnHS\_VerifyResponseEx\_WithInfo() 호출 후 다음과 같은 파라미터 정보를 반환
  - ulError: ERROR\_ANICPXSVR\_DETECT\_CALLBACK\_IS\_NOTIFIED
  - ulSpecificError: AHNHS\_ACTAPC\_DETECT\_ANTIFREESERVER (0x10910)

### 9.3.1 툴 사용 방법

SetServerList 툴을 사용하는 순서는 다음과 같습니다.

1. afs.dat 를 생성합니다.
2. afs.dat 를 배포합니다.

각 순서에 대한 자세한 설명은 다음과 같습니다.

#### afs.dat 파일 생성

[HackShield SDK]\Bin\Win\x86\Util\SetServerList.exe를 사용하여 afs.dat 파일을 생성합니다. 사용 방법은 다음과 같습니다.

SetServerList.exe 사용 방법은 다음과 같습니다.

[HackShield SDK]\Bin\Win\x86\Util\SetServerList.exe

### 그림 58 SetServerList.exe 사용 방법



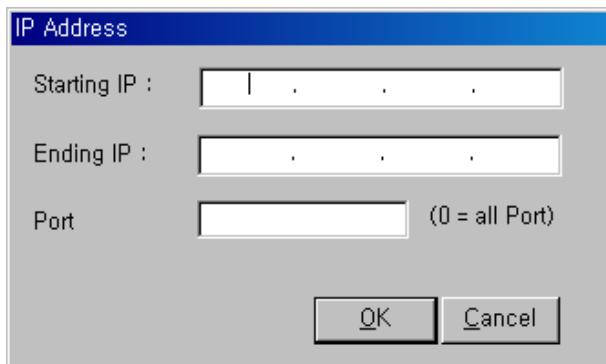
#### 주의

SetServerList.exe는 업데이트 설정 파일을 생성하는 데므로 배포하지 마십시오.

1. SetServerList.exe를 실행합니다.



2. **Insert**를 누르면 IP 주소를 입력하는 팝업 창이 나타나면 유효한 IP 주소의 범위를 지정합니다. 시작 주소와 끝 주소를 입력합니다. 만약 하나의 IP만 추가하려면 시작 주소와 끝 주소를 동일하게 입력합니다.



3. **Port**는 차단 IP 주소 범위(Black-list) 방식의 경우만 입력이 가능합니다. 설정한 주소 범위 내에 해당 Port로 접속한 세션이 존재하면 게임 서버에서 종료할 수 있도록 서버 연동을 통해

ERROR\_ANTICPXSvr\_DETECT\_CALLBACK\_IS\_NOTIFIED 를 서버에 전달합니다. 단, 0 번 포트의 경우는 모든 포트 번호를 대상으로 합니다.

4. **OK** 를 누르면 순서 2에서 입력한 IP 주소가 추가된 것을 확인할 수 있습니다.
5. 리스트에서 IP 주소를 누르거나 선택한 후 **Modify** 를 누르면 수정할 수 있습니다.
6. 리스트에서 IP 주소를 선택하고 **Delete** 를 누르면 해당 주소가 삭제됩니다.
7. **Save** 를 누르면 지정한 IP 주소와 포트 값들이 afs.dat 파일에 저장되고 프로그램이 종료됩니다. 저장하지 않으면 수정 사항이 반영되지 않으므로 반드시 수정 후 저장 버튼을 눌러야 합니다.
8. afs.dat 파일은 배포시 HackShield 모듈과 동일한 경로에 배포해야 합니다. 예를 들어, [Game Directory]/hshield/afs.dat 와 같이 배포합니다.

### afs.dat 파일 배포

afs.dat 파일은 고객이 관리하고 배포하는 것을 원칙으로 합니다. 다만 고객이 afs.dat 파일에 대한 배포가 불가능할 경우를 대비하여 HackShield 업데이트를 통해 배포하도록 지원합니다. HackShield 업데이트를 통해 afs.dat 파일을 배포하는 경우에도 afs.dat 파일은 고객이 관리해야 합니다.

업데이트 서버에서 HackShield 패치셋 하위(ahn.ui, autoup.exe 등과 동일한 경로)에 afs.dat 파일을 위치시키면 HackShield 업데이트 시 자동으로 다운로드합니다.



HackShield 업데이트 구조에 대해서는 [3.1 HackShield 업데이트 기능](#)을 참고하십시오.

---

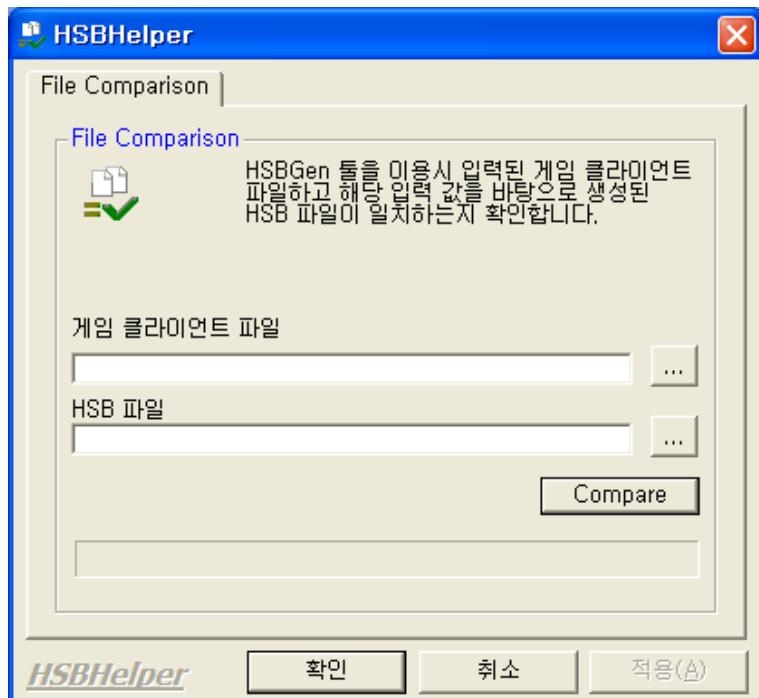
## 9.4 HSBHelper 툴

₩Bin₩Win₩x86₩AntiCrack₩HSBHelper 를 이용하여 HSB 파일과 그에 상응하는 게임 클라이언트 파일이 맞는지 확인할 수 있습니다.

### 9.4.1 툴 사용 방법

HSBHelper 툴을 사용해서 HSB 파일과 그에 상응하는 게임 클라이언트 파일이 맞는지 윈도우 창이나 Command Line에서 확인할 수 있습니다.

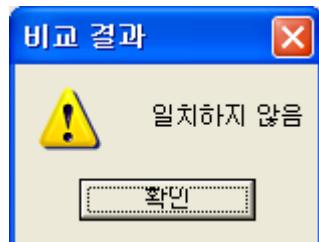
윈도우 창을 사용하여 확인하는 방법은 다음과 같습니다.



1. 게임 클라이언트 파일이 위치한 전체 경로명을 입력합니다. 예를 들어, C:\ R.5.1.41.1(build 671)\Bin\Win\x86\Amazon.exe 와 같이 입력합니다.
2. HSB 파일이 위치한 전체 경로명을 입력합니다. 예를 들어, C:\ R.5.1.41.1(build 671)\Bin\Win\x86\AntiCrack\AntiCpX.hsb 와 같이 입력합니다.
3. Compare 를 누릅니다.
4. 일치하는 경우, 다음과 같은 메시지 박스가 나타납니다.



5. 일치하지 않는 경우에는 다음과 같은 메시지 박스가 나타납니다.



#### 주의

HSBHelper 실행시, HSBHelper.log 로그 파일이 항상 생성됩니다.

## Command Line 실행

Command Line을 사용하여 확인하는 방법은 다음과 같습니다.

HSBHelper.exe (1), (2)

### 그림 59 HSBHelper.exe 사용 방법

(1)에는 HSB 정보를 추출할 게임 경로를 입력하고 (2)에는 HSB 파일의 경로를 입력합니다.

---

#### ⚠ 주의

인자 정보는 '' 단위로 구분됩니다. 결과가 성공인 경우 '0'을 반환하고, 그 외 값은 실패입니다.

---

# **10장**

## **자주 하는 질문**

**질문:** 현재 시스템에 SoftICE가 설치되어 있습니다. HackShield 초기화가 정상적으로 이루어지지 않는데, 어떻게 해야 합니까? 또는 개발 및 디버깅 작업을 하는 경우 초기화 옵션을 어떻게 설정합니까?

**답변:** 개발이나 테스트 과정에서 게임 클라이언트를 디버깅하려면 개발자용 HShield.lib, Ehsvc.dll, hshield.dat를 사용해야 합니다. 단, Ehsvc.dll과 hshield.dat 파일이 릴리즈 버전과 섞이면 서버 연동 시 문제가 발생할 수 있으니 유의해야 합니다.

HackShield Developer 버전 경로는 다음과 같습니다.

- EhSvc.dll 경로: \SDK\Korean(kr)-SDK\Developer\Bin\Win\x86\HShield
- hshield.dat 경로: \SDK\Korean(kr)-SDK\Developer\Bin\Win\x86\HShield  
(서버 연동 적용 시(후) 디버깅할 때 서버에 복사해야 함)
- HShield.lib 경로: \SDK\Korean(kr)-SDK\Developer\Lib\Win\x86\  
(게임 클라이언트 프로젝트의 컴파일 항목에 맞는 적절한 라이브러리 파일 사용)

---

#### 주의

VC++ 컴파일러의 exception 셋팅이 Microsoft C++ Exception: Stop always로 설정되어 있는 경우 다음과 같은 에러가 발생할 수 있습니다.

"First chance exception in Game.exe(KERNEL32.dll) 0xE0607063 Microsoft  
C++ Corporation"

VC++ 컴파일러의 exception 셋팅이 Microsoft C++ Exception: Stop always로 설정되어 있는지 확인한 다음 해당 설정 값을 Stop if not handled로 변경하십시오.

---

**질문:** Windows Vista에서 Manifest 기능을 사용하려고 하면 어떻게 해야 합니까?

**답변:** manifest 파일 내용(xml 형식)을 게임 리소스에 적용하는 순서는 다음과 같습니다.

---

#### 참고

Vista에서는 Administrators 권한이 있는 경우에만 정상적으로 동작될 수 있는 애플리케이션을 만들기 위해서 manifest를 이용해 '이 프로그램을 동작하기 위해서는 반드시 권한상승이 필요하다'는 정보를 실행 파일에 포함해야 합니다. 오른쪽 마우스를 눌러서 admin 권한으로 실행으로 애플리케이션을 동작하거나 속성에서 admin 권한으로 이 프로그램 실행을 선택할 수도 있습니다. 하지만 사용자가 애플리케이션을 동작하는 것은 상당히 번거로운 일이기 때문에 실행 파일 자체에 애플리케이션은 반드시 Administrators permission이 필요하다는 정보를 추가하여, 자동적으로 권한상승이 이루어지도록 하는 것을 권장합니다.

---

1. Visual Studio 6.0 을 실행합니다.
2. 게임 리소스 폴더에 Resource 를 추가합니다.

3. resource 항목의 new custom resource(Resource type)에 24를 입력합니다.

 참고

24는 Microsoft에서 정의한 manifest resource value입니다.

4. 다음의 xml 내용을 순서 3에서 추가된 IDR\_DEFAULT1에 복사합니다.

```
<?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. 복사한 다음 바이너리와 ASCII 코드의 xml 문서 내용을 다음과 같이 확인합니다.

00000000	3C 3F 78 6D 6C 20 76 65	72 73 69 6F 6E 3D 22 31	<?xml version="1
00000100	2E 30 22 20 65 6E 63 6F	64 69 6E 67 3D 22 75 74	.0" encoding="ut
00000200	66 2D 38 22 20 3F 3E 0D	0A 3C 61 73 73 65 6D 62	f-8" ?>..<assemb
00000300	6C 79 20 78 6D 6C 6E 73	3D 22 75 72 6E 3A 73 63	ly xmlns="urn:sc
00000400	68 65 6D 61 73 2D 6D 69	63 72 6F 73 6F 66 74 2D	hemas-microsoft-
00000500	63 6F 6D 3A 61 73 6D 2E	76 31 22 20 6D 61 6E 69	com:asm.v1" mani
00000600	66 65 73 74 56 65 72 73	69 6F 6E 3D 22 31 2E 30	festVersion="1.0
00000700	22 3E 0D 0A 3C 61 73 73	65 6D 62 6C 79 49 64 65	
00000800	6E 74 69 74 79 20 76 65	72 73 69 6F 6E 3D 22 31	nity version="1
00000900	2E 30 2E 30 2E 30 22 20	0D 0A 20 20 20 20 70 72	.0.0." .. pr
00000a00	6F 63 65 73 73 6F 72 41	72 63 68 69 74 65 63 74	ocessorArchitect
00000b00	75 72 65 3D 22 58 38 36	22 0D 0A 20 20 20 20 6E	ure="X86" .. n
00000c00	61 6D 65 3D 22 4D 69 6E	69 41 22 0D 0A 20 20 20	ame="Minia" ..
00000d00	20 74 79 70 65 3D 22 77	69 6E 33 32 22 20 2F 3E	type="win32" />
00000e00	20 0D 0A 20 20 3C 64 65	73 63 72 69 70 74 69 6F	.. <descriptio
00000f00	6E 3E 4D 69 6E 69 41 3C	2F 64 65 73 63 72 69 70	n>Minia</descrip
00010000	74 69 6F 6E 3E 0D 0A 20	20 3C 74 72 75 73 74 49	tion>.. <trustI
00011000	6E 66 6F 20 78 6D 6C 6E	73 3D 22 75 72 6E 3A 73	nfo xmlns="urn:s
00012000	63 68 65 6D 61 73 2D 6D	69 63 72 6F 73 6F 66 74	chemas-microsoft
00013000	2D 63 6F 6D 3A 61 73 6D	2E 76 33 22 3E 0D 0A 20	-com:asm.v3">..
00014000	20 20 20 3C 73 65 63 75	72 69 74 79 3E 0D 0A 20	<security>..
00015000	20 20 20 20 3C 72 65	71 75 65 73 74 65 64 50	<requestedP
00016000	72 69 76 69 6C 65 67 65	73 3E 0D 0A 20 20 20 20	rivileges>..
00017000	20 20 20 20 20 3C 72	65 71 75 65 73 74 65 64	<requested
00018000	45 78 65 63 75 74 69 6F	6E 4C 65 76 65 6C 20 6C	ExecutionLevel 1
00019000	65 76 65 6C 3D 22 72 65	71 75 69 72 65 41 64 6D	evel="requireAdm
0001a000	69 6E 69 73 74 72 61 74	6F 72 22 20 20 2F 3E 0D	inistrator" />.
0001b000	0A 20 20 20 20 20 3C	2F 72 65 71 75 65 73 74	.. <request
0001c000	65 64 50 72 69 76 69 6C	65 67 65 73 3E 0D 0A 20	edPrivileges>..
0001d000	20 20 20 3C 2F 73 65 63	75 72 69 74 79 3E 0D 0A	</security>..
0001e000	20 20 3C 2F 74 72 75 73	74 49 6E 66 6F 3E 0D 0A	</trustInfo>..
0001f000	3C 2F 61 73 73 65 6D 62	6C 79 3E	</assembly>

6. 새로 생성된 리소스 파일 IDR\_DEFAULT1에서 마우스 오른쪽 버튼 누릅니다.

7. properties 메뉴 누른 다음 IDR\_DEFAULT1 을 1로 변경합니다.

---

 참고

24 manifest 기능을 사용하려면 ID 값을 1로 설정합니다.

---

8. 저장한 다음 게임 리소스를 재빌드합니다.

---

 참고

Manifest 기능을 사용해야 하는 이유는 다음과 같습니다. 예를 들어 해커나 일반 유저가 악의의 목적으로 해킹툴을 admin 권한으로 게임을 실행하면 해킹툴은 유저 레벨이 아닌 admin 권한에서 동작합니다. 이 때 게임이 유저 권한에서 동작한다면, 게임은 하위레벨에서 동작하게 되어 해킹툴에 대한 방어가 불가능합니다. 낮은 권한의 애플리케이션은 상위 권한의 애플리케이션에 대한 액세스를 취할 수 없습니다. 또한, Vista에서는 프로세스 실행 중에는 권한 상승을 할 수가 없다는 것도 한 가지 이유입니다.

admin 권한의 해킹툴에 대한 방어를 위해서 HackShield 또한 반드시 admin 권한에서 실행해야 합니다. HackShield는 이러한 이유로 Vista에서 쉐도우 계정 생성이 불필요하기에, manifest 기능의 추가가 필요합니다.

---

### 질문: HackShield Update 서버 구축은 어떻게 합니까?

**답변:** 이 매뉴얼과 함께 제공되는 AhnLab HackShield for Online Game 2.0 서버 퀵 가이드를 참고하십시오.

### 질문: Windows 2000이나 Windows XP에서 Administrator 권한이 아닌 사용자로 시스템에 로그인하면 HackShield를 사용할 수 있습니까?

**답변:** HackShield은 커널 레벨에서 해킹 차단 드라이버를 구동합니다. 따라서 기본 설정 상태에서 Administrator 권한이 없는 사용자는 HackShield를 사용할 수 없습니다. 관리자 계정이 아닌 일반 사용자 계정으로 HackShield의 게임 해킹 방어 기능을 실행하려면 HackShield에서 사용할 쉐도우 계정(shadow account)을 생성하십시오.

### 질문: 게임 서비스 중에서 여러 원인으로 HackShield 종료 함수 \_AhnHS\_StopService(), \_AhnHS\_Uninitialize()가 호출되지 않고 게임이 비정상적으로 종료되었습니다. 게임을 다시 시작하면 HackShield가 다시 실행됩

## 니까?

**답변:** HackShield 종료 함수를 호출하지 않고 게임을 종료하면 해킹 차단 드라이버가 시스템에서 언로드되지 않은 상태가 됩니다. 그러나 이 때 HackShield 초기화 함수를 호출하여 게임을 다시 실행하면 자동으로 기존 드라이버를 강제로 언로드시키고 새 드라이버를 로딩시켜 HackShield가 정상적으로 초기화됩니다.

## 질문: 새롭게 발견한 해킹툴을 차단하려면 어떻게 해야 합니까?

**답변:** 새로운 해킹툴이 발견되면 다음과 같은 정보를 안랩 HackShield 기술 지원에 전달합니다.

- 해킹툴이 동작하는 게임
- 동작 운영체제 버전
- 해킹툴에 대한 간략한 설명
- 해킹툴 실행 파일

전달된 정보를 분석하여 매주 발생하는 정기 엔진 업데이트나 긴급 엔진에 적용하도록 처리됩니다.

## 질문: HackShield가 적용된 상태에서 HackShield 기능을 동작하지 않게 할 수 있는 방법이 있나요?

**답변:** HackShield를 게임에 적용한 상태에서 HackShield의 기본 기능, 업데이트, 서버 연동 기능을 동작하지 않게 할 수 있습니다.

HackShield 라이브러리가 적용된 상태에서 \_NO\_HACKSHIELD 플래그를 전처리합니다. 또는 게임 프로젝트 빌드 셋팅에서 Preprocessor definitions 옵션에 \_NO\_HACKSHIELD 플래그를 추가해도 됩니다. HackShield 기능이 동작하지 않게 하는 경우와 전처리문에 대한 정보는 다음과 같습니다.

**표 23 HackShield 기능 OFF**

Case	전처리문
HackShield 기본 기능 OFF	#define _NO_HACKSHIELD #include "HShield.h"
HackShield 업데이트 기능 OFF	#define _NO_HACKSHIELD #include "HSUpChk.h"

---

HackShield 서버 연동 기능 OFF	#define _NO_HACKSHIELD #include "AntiCpXSvr.h"
-------------------------	---------------------------------------------------

---

**⚠️ 주의**

\_NO\_HACKSHIELD 전처리를 적용하거나 해제한 후에는 반드시 게임을 재빌드해야 합니다. 해당 기능은 5.4.6.1 릴리즈부터 지원합니다.

---

# AhnLab

이메일: <http://www.ahnlab.com>의 고객지원 → 1:1 메일 상담

기업 기술지원 서비스: 1577-9431

팩스: 031-722-8901

주소: (우)463-400 경기도 성남시 분당구 살평동 673 주식회사 안랩

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

AhnLab, the AhnLab logo, and HackShield are trademarks or registered trademarks of AhnLab, Inc., in Korea and certain other countries. All other trademarks mentioned in this document are the property of their respective owners.