

AhnLab

HackShield for Online Game 2.0



클라이언트 퀵 가이드

AhnLab

일러두기

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

이 클라이언트 퀵 가이드의 내용과 프로그램은 저작권법과 컴퓨터프로그램보호법에 의해서 보호받고 있습니다.



면책 조항

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

이 클라이언트 퀵 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 클라이언트 퀵 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), 고객 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 HackShield 클라이언트에 대한 다양한 기능 설정 및 적용 방법을 설명한 퀵 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

이 퀵 가이드는 Microsoft Visual C++ 6.0 환경에서 C/C++ 언어 사용을 기준으로 작성했습니다. 대략적인 설명과 간략한 샘플 코드를 다루기 때문에 세부 구현은 각 고객의 정책에 맞추어 진행하십시오.

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
- 전화: 1577-9431
- 팩스: 031-722-8901

목차

1장 기본 기능	7
1.1 HackShield 소개	8
1.2 HackShield 기본 기능	8
1.2.1 특징	8
1.2.2 구성	8
2장 적용	9
2.1 적용 준비	10
2.2 HackShield 기본 기능 적용	10
2.3 적용 시 주의사항	18
3장 테스트 및 배포	19
3.1 테스트	20
3.2 배포	20

그림 목차

그림 1	헤더 파일 선언 예제	10
그림 2	_AhnHS_Initialize() 호출 예제	12
그림 3	_AhnHS_StartService() 호출 예제	14
그림 4	_AhnHS_StopService() 호출 예제	14
그림 5	_AhnHS_Uninitialize() 호출 예제	15
그림 6	HackShield 종료 루틴 수행	15
그림 7	_AhnHS_Callback() 호출 예제	16

표 목차

표 1 HackShield 기본 기능 파일 구성	8
----------------------------------	---

1장 기본 기능

1.1 HackShield 소개 /8

1.2 HackShield 기본 기능 /8

1.1 HackShield 소개

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

1.2 HackShield 기본 기능

HackShield가 온라인 게임 해킹 방지를 위해 다양한 기능을 제공합니다.

1.2.1 특징

HackShield 기본 기능의 특징은 다음과 같습니다.

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

1.2.2 구성

HackShield 기본 기능을 위한 파일 구성은 다음과 같습니다.

표 1 HackShield 기본 기능 파일 구성

파일 경로	설명
\Bin	HackShield 실행에 관련된 파일
\Data	HackShield에서 사용하는 데이터 파일
\Developer	HackShield 적용 시 개발자 테스트를 위해 필요한 파일
\Doc	사용 가이드
\Include	HackShield 적용을 위한 인클루드 파일
\Lib	HackShield 적용을 위한 라이브러리 파일
\PatchSet	업데이트를 위한 패치셋 파일
\Sample	사용자 지원을 위한 샘플 파일

2장 적용

2.1 적용 준비 /10
2.2 HackShield 기본 기능 적용 /10
2.3 적용 시 주의사항 /18

2.1 적용 준비

HackShield 기본 기능을 적용하기 전에 준비해야 하는 사항은 다음과 같습니다.

- 라이브러리 추가
 - Windows 기본 라이브러리 `version.lib`, `winmm.lib` 를 프로젝트 추가
 - `\Lib\HShield.lib` 를 적당한 경로에 복사한 다음 프로젝트에 추가
- 헤더 파일 추가
 - `\Include\HShield.h` 를 적당한 경로에 복사한 다음 프로젝트에 추가
- 라이선스 키 발급
 - 게임 코드(4 자리 숫자)와 24 자리 문자열로 구성
 - 각 게임마다 고유한 값으로 발급

2.2 HackShield 기본 기능 적용

HackShield 기본 기능을 적용하는 순서는 다음과 같습니다.

1. 헤더 파일 선언
2. 초기화(Initialize)
3. 시작(Start)
4. 게임 실행
5. 정지(Stop)
6. 종료(Uninitialize)

게임 클라이언트의 초기화 이전에 HackShield 모듈의 초기화와 시작을 수행하고 종료 시에는 반드시 정지와 종료를 수행하십시오.

적용 순서에 대한 자세한 설명은 다음과 같습니다.

헤더 파일 선언

게임 프로젝트에 복사한 HShield.h 파일을 위치에 맞게 선언합니다.

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

그림 1 헤더 파일 선언 예제

HackShield 호출

게임의 엔트리-포인트 함수에 HackShield를 호출하는 함수를 추가합니다.

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

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

    // 게임 실행 루틴 부분
    ...

    HS_StopService();         HackShield Service Stop 부분
    HS_UnInit();               HackShield Uninitialize 부분
}
```

Uninitialize가 제대로 호출되지 않으면 코드를 다시 수행할 때 정상적으로 실행되지 않을 수 있습니다. Initialize(Start)를 실패했을 때 처리하는 부분과 게임 실행 중 Exception을 처리하는 부분에 반드시 Uninitialize(Stop)를 호출하도록 처리하십시오.

게임 클라이언트의 안전한 보호를 위하여 HackShield의 초기화가 끝나면 바로 HackShield를 시작하는 것을 권장합니다.

초기화(Initialize)

HackShield 초기화 함수 `_AhnHS_Initialize()` 호출 예제는 다음과 같습니다.

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

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

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

```

// ③ _AhnHS_Initialize()를 호출하여 HackShield를 초기화 합니다.
nRet = _AhnHS_Initialize (
    szFullFilePath,
    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 (
//    szFullFilePath,
//    HS_CallbackProc,          // 콜백 함수
//    1000,                    // 게임 코드
//    "B228F291B7D7FAD361D7A4B7", // 라이선스 키
//    dwOption,                // 옵션 플래그
//    unAdditionalRatio );
//

// ④ _AhnHS_Initialize()의 리턴 값을 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 2 _AhnHS_Initialize() 호출 예제

위 예제에 대한 자세한 설명은 다음과 같습니다.

① `szFullFilePath`에는 HackShield SDK의 `EhSvc.dll`의 절대 경로를 설정합니다. 이 파일은 일반적으로 실행 파일 경로에 생성될 HShield 폴더 아래에 위치합니다. 올바른 절대 경로 설정을 위해서 `GetModuleFileName()`을 이용하십시오. `GetCurrentDirectory()`는 원하는 경로를 얻지 못할 수 있습니다.

② 여기서 정의한 플래그는 `_AhnHS_Initialize()` 호출 시 5번째 인자로 사용합니다. HackShield의 기본적인 기능을 설정합니다. 게임 디버깅을 위해서 몇 개의 옵션을 삭제하고 실행해야 합니다. 특별한 이슈가 없는 경우 테스트 환경 및 배포 시 위에서 사용한 플래그를 그대로 사용하십시오.

③ `_AhnHS_Initialize()` 호출하려면 게임 코드와 라이선스 키를 발급 받아야 합니다. 고객 정책에 따라 마지막 인자에 스피드 해 감지 민감도와 핵실프 자체 종료 대기시간을 설정합니다.

일반적으로는 스피드 해 감지 민감도는 `AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL`를 사용하고, `AHNHS_CHKOPT_SELF_DESTRUCTION` 옵션이 적용되었다면 기본적으로 `AHNHS_SELFDESTRUCTION_RATIO_NORMAL`이 적용됩니다.

④ `_AhnHS_Initialize()`의 반환값을 받아서 정상 수행(`HS_ERR_OK`)이 아니라면 해당하는 에러 코드에 따라 에러 메시지를 출력하고 종료합니다. 에러 코드 `HS_ERR_INVALID_FILES`의 에러 메시지 예제는 다음과 같습니다.

```
wsprintf( szMsg, "잘못된 파일이 설치되었습니다.\n프로그램을 재설치 하시기 바랍니다. (%x)", nRet );
break;
```

❗ 참고

각 플래그 및 `_AhnHS_Initialize()`의 각 인자와 에러 메시지에 대한 자세한 내용은 AhnLab HackShield for Online Game 2.0 프로그래밍 가이드를 참고하십시오.

시작(Start)

HackShield 시작 함수 `_AhnHS_StartService()` 호출 예제는 다음과 같습니다.

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

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

    // ② _AhnHS_StartService()의 리턴 값을 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 3 _AhnHS_StartService() 호출 예제

HackShield를 시작하기 위해서는 반드시 _AhnHS_Initialize()가 호출되어야 합니다. 실제 해킹에 대한 방어가 이루어지는 것은 _AhnHS_StartService()가 호출된 이후이므로 게임 초기화 루틴을 수행하기 전에 HackShield 시작을 수행하는 것이 좋습니다.

_AhnHS_StartService()의 리턴 값을 받아서 정상 수행(HS_ERR_OK)이 아니라면 해당하는 에러 코드에 따라 적당한 에러 메시지를 출력하고 프로그램을 종료합니다.

각 경우에 대한 에러 메시지는 AhnLab HackShield for Online Game 2.0 프로그래밍 가이드를 참고하여 특성에 맞도록 작성합니다.

정지(Stop)

HackShield 정지 함수 _AhnHS_StopService() 호출 예제는 다음과 같습니다.

```

BOOL HS_StopService()
{
    int nRet = HS_ERR_OK;

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

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

    return TRUE;
}

```

그림 4 _AhnHS_StopService() 호출 예제

HackShield를 정지하기 위해서는 HackShield가 동작 중이어야 합니다. _AhnHS_StopService()를 호출하면 HackShield가 정지되어 해킹 차단 기능과 해킹툴 탐지 기능을 정지시킵니다.

종료(Uninitialize)

HackShield 종료 함수 `_AhnHS_Uninitialize()` 호출 예제는 다음과 같습니다.

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

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

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

    return TRUE;
}
```

그림 5 `_AhnHS_Uninitialize()` 호출 예제

게임 클라이언트 프로그램이 비정상적으로 종료되면 HackShield 해킹 차단 드라이버를 언로드하지 못할 수 있습니다. 이 경우 다시 실행할 때 정상적으로 실행되지 않을 수 있으므로 서비스를 종료할 때에는 반드시 Uninitialize를 수행하도록 합니다. 클라이언트의 정상 종료는 물론, 콜백 함수 호출 후 종료와 비정상 종료되는 exception처리에서도 stop과 Uninitialize를 반드시 수행하십시오.

비정상 종료 상황에 대비하여 게임의 최초 시작 위치에 다음과 같은 코드를 추가하면 예외로 인한 게임 종료 시 HackShield 종료 루틴을 수행할 수 있습니다.

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

그림 6 HackShield 종료 루틴 수행

`Game_UnhandledExceptionHandler()` 내부에 HackShield StopService와 Uninitialize를 호출하도록 처리합니다. 물론 이 경우는 예외가 발생한 경우에 해당하며, 예외를 발생하지 않고 종료하는 경우는 해당되지 않습니다.

이벤트 전달

해킹 차단 기능과 해킹툴 탐지 기능에 관련된 이벤트 전달 함수 `_AhnHS_Callback()` 호출 예제는 다음과 같습니다.

```
int __stdcall HS_CallbackProc ( long lCode, long lParamSize, void* pParam )
{
    TCHAR szMsg[MAX_PATH];
    // ① 각 경우에 대하여 알맞은 예러 메시지를 출력합니다.
    switch ( lCode )
    {
        // ② Engine Callback
```

```

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 별도의 처리를 하지 않음
case AHNHS_ACTAPC_DETECT_AUTOMOUSE:
case AHNHS_ACTAPC_DETECT_ALREADYHOOKED:
break;
// ⑤ Speedhack 관련
case AHNHS_ACTAPC_DETECT_SPEEDHACK:
    wsprintf( szMsg, "스피드해킹으로 의심되는 동작이 감지되었습니다.
    (%x)", lCode );
    MessageBox( NULL, szMsg, szTitle, MB_OK );
    break;
// ⑥ 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;
// 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_AHNHS_ACTAPC_DETECT_ENGINEFAILED:
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 7 _AhnHS_Callback() 호출 예제

위 예제에 대한 자세한 설명은 다음과 같습니다.

①HackShield가 감지한 정보를 처리합니다. 게임 정책에 맞게 각 경우에 대해 정의하십시오.

참고

각 경우에 대한 자세한 내용은 AhnLab HackShield for Online Game 2.0 프로그래밍 가이드를 참고하십시오.

②해킹 프로그램이거나 게임 진행에 문제를 일으킬 가능성이 있는 프로그램을 감지한 경우입니다. pParam을 통해 감지된 프로그램의 이름을 알 수 있으므로 게임 유저에게 알려줍니다. 메시지가 발생시킨 후 게임 클라이언트를 계속 진행할 수 없도록 종료하십시오.

③HackShield에서 오토 매크로와 관련된 해킹툴을 탐지한 경우입니다. 게임 클라이언트를 계속 진행할 수 없도록 종료하십시오.

④별도의 메시지 박스나 에러 처리를 하지 않는 것이 좋습니다. 필요한 경우 로그만 작성합니다. HNHS_ACTAPC_DETECT_ALREADYHOOKED의 경우 일부 API가 이미 후킹되어있는 경우이며 보안 프로그램과 같이 정상적인 프로그램에서도 후킹할 수 있습니다. AHNHS_ACTAPC_DETECT_AUTOMOUSE의 경우도 Detection이 아닌 Protection 기능이므로 해당 콜백 함수가 호출되지 않습니다.

⑤스피드 핵과 같이 시스템의 시간 변화 속도가 비정상적인 경우입니다. 스피드 핵일 가능성이 높기 때문에 메시지를 발생시키고 고객 정책에 따라 종료 여부를 판단하십시오.

⑥디버깅 시도를 감지한 경우입니다. 게임 프로그램에 대한 디버깅 작업을 진행 중일 가능성이 높으므로 메시지를 발생시킨 후 게임 클라이언트를 계속 진행할 수 없도록 종료하십시오.

⑦메시지 후킹이나 모듈 변경과 같은 이상이 생긴 경우입니다. 메시지를 발생시킨 후 게임 클라이언트를 계속 진행할 수 없도록 종료하십시오.

HackShield 기능 적용 해제

HackShield를 게임에 적용한 상태에서 HackShield의 기본 기능, 업데이트 기능, 서버 연동 기능을 동작하지 않게 할 수 있습니다.

HackShield 라이브러리가 적용된 상태에서 전처리기 정의에 _NO_HACKSHIELD를 추가합니다. 게임 프로젝트 빌드 셋팅에서 Preprocessor definitions 옵션에 _NO_HACKSHIELD 플래그를 추가하는 방법도 있습니다. 각 경우를 정리하면 다음과 같습니다.

■ Case 1: HackShield 기본 기능 OFF

```
#define _NO_HACKSHIELD
#include "HShield.h"
```

■ Case 2: HackShield 업데이트 기능 OFF

```
#define _NO_HACKSHIELD
#include "HSUpChk.h"
```

■ Case 3: HackShield 서버 연동 기능 OFF

```
#define _NO_HACKSHIELD
#include "AntiCpXSvr.h"
```

주의

_NO_HACKSHIELD 전처리를 적용 혹은 해제한 후에는 반드시 게임을 다시 빌드하십시오. 해당 기능은 5.4.6.1 릴리즈부터 지원합니다.

2.3 적용 시 주의사항

적용 시 주의해야 할 사항은 다음과 같습니다.

Initialize 예제에서 사용한 옵션

예제에서 Initialize 할 때 사용한 옵션은 고객의 정책에 따라 선택적으로 적용할 수 있습니다. 안전한 게임 클라이언트의 보호를 위해서 예제에서 사용된 옵션을 그대로 사용하는 것을 권장합니다.

에러 메시지 출력

디버깅 및 고객 지원을 위해 에러 메시지를 출력할 때 에러 코드를 함께 출력하는 것을 권장합니다.

메시지 처리

메시지 처리 예제에서 `MessageBox(NULL, ...)`를 사용했으나 실제 Full-Screen 게임에서는 게임화면 뒤로 나타날 수 있으므로 가능하면 게임 상의 메시지로 처리하십시오. `MessageBox()`를 사용한다면, `NULL` 대신 `Handle` 처리를 해야 합니다.

에러 발생 시 메시지만 나타나면 에러를 처리하지 않고 메시지를 무시한 채 게임을 계속 진행할 수 있습니다. 에러가 발생하면 반드시 클라이언트와의 접속을 종료하고 해당 클라이언트의 소켓 종료와 접속 리스트에서 제거하는 작업을 수행하십시오.

3장

테스트 및 배포

3.1 테스트 /20

3.2 배포 /20

3.1 테스트

작성한 소스 코드를 테스트합니다. 테스트하는 순서는 다음과 같습니다.

1. HackShield 코드의 적용이 완료된 프로젝트를 컴파일하여 게임 클라이언트를 생성합니다.
2. 게임 클라이언트 폴더 아래에 다음과 같이 HShield 라는 폴더를 생성합니다.

```
...\[Game Directory]\HShield
```

3. 제공되는 \Bin\HShield 폴더의 파일들을 앞서 생성한 HShield 폴더 안에 복사합니다.
4. 게임 클라이언트를 실행합니다.
5. 다음과 같은 테스트로 HackShield 동작 여부를 판단합니다.
 - Process Explorer 실행하여 dll 정보가 나타나는지 확인
 - 게임 실행 전 HShield 폴더 내 EhSvc.dll 파일을 rename 하고 이를 감지하는지 확인
 - 해킹툴을 동작시켜서 게임 클라이언트에서 이를 정확하게 감지하는지 확인
6. 이상이 없다면 게임 실행 파일을 패키징합니다.

참고

패커(Packer)를 사용하면 게임 실행 파일이 쉽게 디버깅되는 것을 방지할 수 있습니다. HackShield는 upx 패커를 default로 제공합니다. upx 패커 사용 예제는 다음과 같습니다.
upx.exe -f Source.exe -o Output.exe

참고

이상이 없을 경우, 서버 연동 적용을 권장합니다. 서버 연동 적용 방법은 AhnLab HackShield for Online Game 2.0 서버 퀵 가이드를 참조하십시오.

3.2 배포

테스트가 끝난 다음 최초로 배포하는 순서는 다음과 같습니다.

1. 적용 및 테스트가 끝나면 안랩에서 QA(Quality Assurance)를 진행합니다.
2. QA Report 를 전달 받아 수정할 사항을 확인합니다.
3. 수정이 끝난 최종 배포 버전을 작성합니다.
4. 작성한 최종 배포 버전을 게임 유저에게 배포합니다.

주의

테스트 도중 생성된 hshield.log 파일은 삭제하고 배포하십시오.

AhnLab

이메일: <http://www.ahnlab.com>의 고객센터 → 1:1 메일 상담

기업 기술지원 서비스: 1577-9431

팩스: 031-722-8901

주소: (우)463-400 경기도 성남시 분당구 삼평동 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.