AhnLab HackShield

프로그래밍 가이드

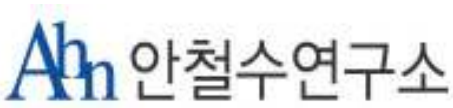
Version 5.1

AhnLab HackShield Pro for Online Game

Quick Guide

V 2.0

기본 기능



1 시작하기 전에

- 이 문서는 HackShield Pro(이하 핵쉴드) SDK 적용을 위한 가이드라인입니다. Microsoft Visual C++ 6.0 환경에서 C/C++ 언어로 작성하는 것을 기준으로 작성되었습니다. 핵쉴드의 기본 기능에 대한 부분을 다룹니다.
- 처음 핵쉴드 코드 적용을 돕기 위하여 대략적인 설명과 간략한 샘플 코드를 다룹니다. 세부 구현은 각 게임 개발사의 정책에 맞추어 진행하십시오.
- \Sample 폴더 안에 MiniA 라는 샘플 코드가 제공됩니다. 코드 적용에 참고하십시오.
- \Doc 폴더 안에 *"AhnLab HackShield Pro 프로그래밍 가이드"* 문서가 제공됩니다. 이 문서는 HackShield Pro의 전체 구조와 기능 및 API 함수 사용법을 설명한 프로그래밍 가이드입니다. 항목별 자세한 사항은 이 문서를 참고하십시오.

2 HackShield Pro SDK의 구성

- \Bin : 핵쉴드 실행에 관련된 파일.
- \Doc : 사용 설명서.
- \Data : 핵쉴드에서 사용하는 데이터 파일.
- \Developer : 핵쉴드 적용 시 개발자 테스트를 위해 필요한 파일.
- \Include : 핵쉴드 적용을 위한 인클루드 파일.
- \Lib : 핵쉴드 적용을 위한 라이브러리 파일.
- \PatchSet : 업데이트를 위한 패치셋 파일.
- \Sample : 사용자 지원을 위한 샘플 파일.

3 적용 전 준비사항

- 라이브러리를 추가합니다.
  - ✓ 핵쉴드에서는 Windows 기본 라이브러리인 version.lib, winmm.lib를 사용합니다. 이 두 라이브러리 파일을 프로젝트에 추가합니다.
  - ✓ \Lib\HShield.lib 라이브러리 파일이 필요합니다. 적당한 위치에 복사한 다음 프로젝트에 추가합니다.
- \Include\HShield.h 헤더 파일이 필요합니다. 적당한 위치에 복사합니다.
- 라이선스 키가 필요합니다. 이것은 게임 코드(4자리 숫자)와 24자리 문자열 라이선스 키로 구성됩니다. 이 라이선스 키는 각 게임마다 고유한 값으로 발급됩니다.

4 핵쉴드 서비스의 적용

- 게임 프로젝트에 앞서 복사한 HShield.h 파일을 위치에 맞게 인클루드시킵니다.

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

- 게임의 엔트리-포인트 함수에 핵쉴드 서비스를 호출하는 함수를 추가합니다.
  - ✓ 핵쉴드 코드의 적용은 초기화(Initialize) → 시작(Start) → 게임 실행 → 정지(Stop) →

완료(Uninitialize)의 순서로 진행됩니다. 게임 클라이언트의 초기화 이전에 핵쉴드 모듈의 초기화와 시작을 수행하고 종료 시에는 반드시 정지와 완료를 수행하십시오.

```
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 핵쉴드의 초기화가 끝나면 바로 핵쉴드 서비스를 시작하시는 것이 좋습니다.

❖ 핵쉴드 초기화 함수(_AhnHS_Initialize)를 작성합니다.

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

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

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

    // ③ _AhnHS_Initialize 함수를 호출하여 핵쉴드 서비스를 초기화 합니다.
    nRet = _AhnHS_Initialize ( szFullFilePath,
```

```
                                HS_CallbackProc,            // 콜백함수
                                1000,                       // 게임 코드
                                "B228F291B7D7FAD361D7A4B7",    // 라이선스 키
                                dwOption,                   // 옵션 플래그
                                AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL );

    // ④ _AhnHS_Initialize 함수의 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;
}
```

① szFullFilePath 변수에는 핵쉴드 SDK의 EhSvc.dll 파일의 절대 경로를 설정합니다. 이 파일은 일반적으로 실행 파일 경로에 생성될 HShield 폴더 아래에 위치합니다. 올바른 절대 경로의 설정을 위해서 GetModuleFileName 함수를 이용하는 것이 좋습니다. GetCurrentDirectory 함수는 원하는 경로를 얻지 못할 수도 있습니다.

② 여기서 정의한 플래그는 _AhnHS_Initialize 함수를 호출할 때 5번째 인자로 사용됩니다. 핵쉴드의 기본적인 기능을 설정합니다. 게임 디버깅을 위해서는 몇몇 옵션을 삭제하고 실행시켜야 합니다. 각 플래그에 대한 자세한 사항은 *"AhnLab HackShield 프로그래밍 가이드: 2.3. Application Programming Interface"*를 참고하십시오. 특별한 이슈가 없다면 테스트 환경 및 배포 시에 위에서 쓰인 플래그를 그대로 사용하는 것을 권장합니다.

③ _AhnHS_Initialize 함수 호출에는 발급받은 게임 코드와 라이선스 키를 필요로 합니다. 마지막 인자는 스피드 핵 감지에 대한 민감도를 설정합니다. 스피드 핵 감지 정도는 게임사의 정책에 따라서 결정됩니다. 일반적으로는 AHNHS_SPEEDHACK_SENSING_RATIO_NORMAL을 사용합니다. 각 인자에 대한 자세한 사항은 *"AhnLab HackShield 프로그래밍 가이드: 2.3. Application Programming Interface"*를 참고하십시오.

④ _AhHS_Initialize 함수의 리턴 값을 받아서 정상 수행(HS_ERR_OK)이 아니라면 해당하는 에러 코드에 따라 에러 메시지를 출력하고 종료합니다. 각 경우에 대한 에러 메시지는 *"AhnLab HackShield 프로그래밍 가이드"*를 참고하여 에러의 특성과 각 게임 개발사의 정책에 맞도록 작성합니다.

ex) HS_ERR_INVALID_FILES의 경우:

wsprintf(szMsg, "잘못된 파일이 설치되었습니다.\n프로그램을 재설치하시기 바랍니다. (%x)", nRet);

break;

❖ 핵쉴드 서비스 시작 함수(_AhnHS_StartService)를 작성합니다.

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

    // ① _AhnHS_StartService 함수를 호출하여 핵쉴드 서비스를 시작합니다.
    nRet = _AhnHS_StartService();

    // ② _AhnHS_StartService 함수의 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;
}
```

① 핵쉴드 서비스를 시작하기 위해서는 반드시 Initialize 함수가 호출되어야 합니다. 실제 해킹에 대한 방어가 이루어지는 것은 StartService 함수가 호출된 이후이므로 게임 초기화 루틴을 수행하기 전에 핵쉴드의 서비스 시작을 수행하는 것이 좋습니다.

② _AhHS_StartService 함수의 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 프로그래밍 가이드"*를 참고하여 특성에 맞도록 작성합니다.

❖ 핵쉴드 서비스 정지 함수를 작성합니다. (_AhnHS_StopService)

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

```
    // ① _AhnHS_StopService 함수를 호출하여 핵쉴드 서비스를 정지합니다.
    nRet = _AhnHS_StopService();

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

① 핵쉴드 서비스를 정지하기 위해서는 핵쉴드 서비스가 동작 중이어야 합니다. _AhnHS_StopService 함수를 호출하면 핵쉴드 서비스가 정지되어 해킹 차단 기능과 해킹 툴 탐지 기능을 정지시킵니다.

❖ 핵쉴드 서비스 종료 함수를 작성합니다. (_AhnHS_Uninitialize)

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

    // ① _AhnHS_Uninitialize 함수를 호출하여 핵쉴드 서비스를 종료합니다.
    nRet = _AhnHS_Uninitialize();

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

① 게임 클라이언트 프로그램이 비정상적으로 종료되면 핵쉴드 해킹 차단 드라이버를 언로드하지 못 할 수도 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.

② 비정상 종료 상황에 대비하여 게임의 최초 시작 위치에 아래의 코드를 추가하면 예외로 인한 게임 종료 시 핵쉴드 종료 루틴을 수행할 수 있습니다.

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

③ 위에 등록된 Game_UnhandledExceptionHandler() 함수 내부에 핵쉴드 StopService와 Uninitialize를 호출하도록 처리하면 됩니다. 물론 이 경우는 예외가 발생한 경우에 해당하며 예외를 발생하지 않고 종료되어버리는 경우에는 해당되지 않습니다.

❖ 해킹 차단 기능과 해킹 툴 탐지 기능에 관련된 이벤트 전달 함수를 작성합니다. (_AhnHS_Callback)

```
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 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 오토메크로 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 Speed 관련
    case AHNHS_ACTAPC_DETECT_SPEEDHACK:
        wsprintf( szMsg, "스피드핵으로 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;
}
```

① 핵쉴드에서 감지된 정보에 대한 처리를 담당합니다. 게임 룰에 맞도록 각 경우에 대하여 정의하십시오. 각 경우에 대한 세부 내용은 *"AhnLab HackShield 프로그래밍 가이드: 2.3. Application Programming Interface"*를 참고하시기 바랍니다.

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

③ 핵쉴드에서 오토매크로 관련 해킹툴을 탐지한 경우입니다. 게임 클라이언트를 계속 진행할 수 없도록 종료하시기 바랍니다.

④ 이 경우에 대해서는 별도의 메시지 박스나 에러 처리를 하지 않는 것이 좋습니다. 필요에 따라서 로그만 작성합니다. AHNHS_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 Callback이 호출되지 않습니다.

⑤ 스피드 핵과 같이 시스템의 시간 변화 속도가 비정상적인 경우입니다. 스피드 핵일 가능성이 많으므로 메시지를 발생시키고 게임 개발사의 정책에 따라 종료 여부를 판단합니다.

⑥ 디버거 트레이스가 발생한 경우입니다. 게임 프로그램에 대한 디버깅 작업을 진행 중일 가능성이 높으므로 메시지를 발생시킨 후 게임 클라이언트를 계속 진행할 수 없도록 종료하시기 바랍니다.

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

5 적용시 주의사항

- 예제에서 Initialize할 때 사용되는 옵션은 게임 개발사의 정책에 따라 선택적으로 적용 가능합니다. 안전한 게임 클라이언트의 보호를 위해서 예제에서 사용된 옵션을 그대로 사용하는 것을 권장합니다.
- 디버깅 및 고객지원을 위하여 에러 메시지를 출력할 때는 에러 코드를 함께 출력하는 것이 좋습니다.
- 메시지 처리 시 예제로 MessageBox(NULL, …)을 사용하였으나 실제 Full-Screen 게임에서는 게임화면 뒤로 나타날 수 있으므로 가능하면 게임 상의 메시지로 처리하십시오. MessageBox 함수를 사용한다면, NULL 대신 Handle 처리를 해야 합니다.
- 메시지 처리 시 단순히 메시지만 발생시키면 그 상태로 게임을 계속 진행하는 경우가 있으므로 메시지를 발생시키고 반드시 게임 클라이언트를 강제로 종료시키십시오.

6 테스트 및 배포

- 핵쉴드 코드가 적용 완료되었으면 프로젝트를 컴파일하여 게임 클라이언트를 생성합니다.
- 게임 클라이언트 폴더 아래에 HShield라는 폴더를 생성합니다.
  - ✓ ex) …\[Game Directory]\HShield
- 제공되는 \Bin\HShield 폴더의 파일들을 앞서 생성한 HShield 폴더 안에 복사합니다.
- 게임 클라이언트를 실행하고 간단한 테스트를 통하여 핵쉴드 동작 여부를 판단할 수 있습니다.

- ✓ Case 1. Process Explorer를 실행하여 dll 정보가 나타나는지 확인
- ✓ Case 2. HShield 내 EhSvc.dll 파일을 리네임하고 이를 감지하는지 확인
- ✓ Case 3. 해킹 툴을 동작시켜서 게임 클라이언트에서 이를 정확하게 감지하는지 확인

❖ 이상이 없다면 게임 실행 파일을 패킹합니다.

- ✓ 패커(Packer)를 사용하여 게임 실행 파일이 쉽게 디버깅되는 것을 방지할 수 있습니다.
- ✓ 핵쉴드는 기본적인 upx 패커를 제공합니다.
- ✓ ex) upx.exe –f Source.exe –o Output.exe

❖ 이상이 없다면 서버연동을 적용하는 것을 권장합니다(서버연동 Quick Guide 참조).

❖ 최초 배포 시에는 다음 과정에 따라 배포를 진행하십시오.

- ✓ 적용 및 테스트 ⇒ Quality Assurance 진행(Ahnlab) ⇒ QA Report를 전달받은 후 수정 사항 확인 후 최종 배포 버전 작성 ⇒ 배포
- ✓ 테스트 도중 생성된 hshield.log 파일은 삭제하고 배포하는 것이 좋습니다.