

AhnLab

HackShield for Online Game 2.0



서버 퀵 가이드

AhnLab

일러두기

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

이 서버 킷 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 서버 킷 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 서버 킷 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	게임사(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 HackShield의 서버 연동 기능에 대한 소개와 설정 및 적용 방법을 설명한 킷 가이드입니다.

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

이 킷 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 서버 연동 기능	8
1.2.1 특징	8
1.2.2 구성	8
1.2.3 동작	9
2장 적용	11
2.1 적용 준비	12
2.2 서버 적용	14
2.3 클라이언트 적용	19
2.4 적용 시 주의사항	21
3장 테스트 및 배포	22
3.1 테스트	23
3.2 배포	23

그림 목차

그림 1	HackShield 서버 연동 기능의 동작	9
그림 2	HSBGen.exe로 hsb 파일 생성	13
그림 3	헤더 파일 선언 예제	14
그림 4	_AhnHS_CreateServerObject() 호출 예제	15
그림 5	_AhnHS_CreateClientObject() 호출 예제	15
그림 6	_AhnHS_MakeRequest() 호출 예제	16
그림 7	AHNHS_TRANS_BUFFER 구조체	16
그림 8	_AhnHS_VerifyResponse() 호출 예제	17
그림 9	_AhnHS_VerifyResponseEx() 호출 예제	18
그림 10	ANTICPX_RECOMMAND_CLOSE_SESSION 반환값	19
그림 11	_AhnHS_CloseClientHandle() 호출 예제	19
그림 12	_AhnHS_CloseServerHandle() 호출 예제	19
그림 13	_AhnHS_MakeRequest() 호출 예제	20

표 목차

표 1 HackShield 서버 연동 기능 파일 구성	8
-------------------------------------	---

1장 기본 기능

1.1 HackShield 소개 /8

1.2 HackShield 서버 연동 기능 /8

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.

1.2 HackShield 서버 연동 기능

서버 연동(AntiCpX)은 파일/메모리 조작을 감지하여 메모리 상에 로드된 코드 영역 전체를 보호합니다.

1.2.1 특징

HackShield 서버 연동 기능의 특징은 다음과 같습니다.

- HackShield 서버 연동 통신 구조는 기존의 게임에 구현된 통신 구조를 그대로 사용
 - 별도의 통신 모듈 구현이 필요 없는 구조
- 게임 클라이언트 버전을 체크하여 부정 실행 방지
 - 허용하지 않는 버전인 경우, 최신 패치 없이 게임하지 않도록 연결 강제 종료
- 게임 클라이언트의 실행 파일, 메모리, HackShield 모듈에 대한 크랙 여부 판단
 - 정책에 따라 일정한 시간 간격으로 요청메시지와 이에 대한 응답메시지 확인
 - 요청메시지에 대한 응답메시지가 유효하지 않다면 해킹으로 판단하여 접속 종료
 - 응답메시지가 전송되지 않는 경우도 해킹으로 판단하여 접속 종료

1.2.2 구성

HackShield 서버 연동 기능을 위한 파일 구성은 다음과 같습니다.



참고

Solaris용 파일은 32bit x86 플랫폼만 지원합니다.

표 1 HackShield 서버 연동 기능 파일 구성

파일 경로 및 이름	설명
\Bin\Win\[Platform]\AntiCrack\AntiCpXSvr.dll	서버 연동 dll 파일
\Bin\Win\x86\AntiCrack\HSBGen.exe	.hsb 파일 생성 툴
\Bin\Win\x86\AntiCrack\HSPub.key	서버연동 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 (Windows용 32Bit)
\\Lib\\Win\\x64\\AntiCpXSvr.lib	서버 연동 라이브러리 파일 (Windows용 64Bit)
\\Bin\\Linux\\x86\\AntiCrack\\ibanticpxsvr.so	서버 연동 동적 라이브러리 파일 (Linux용 32Bit)
\\Bin\\Linux\\x64\\AntiCrack\\ibanticpxsvr.so	서버 연동 동적 라이브러리 파일 (Linux용 64Bit)
\\Bin\\Solaris\\x86\\AntiCrack\\libanticpxsvr.so	서버 연동 동적 라이브러리 파일 (Solaris용)
\\Lib\\Linux\\x86\\AntiCrack\\libanticpxsvr_st.a	서버 연동 정적 라이브러리 파일 (Linux용 32Bit)
\\Lib\\Solaris\\x86\\AntiCrack\\libanticpxsvr_st.a	서버 연동 정적 라이브러리 파일 (Solaris용)

1.2.3 동작

HackShield 서버 연동 기능의 동작 과정은 다음과 같습니다.

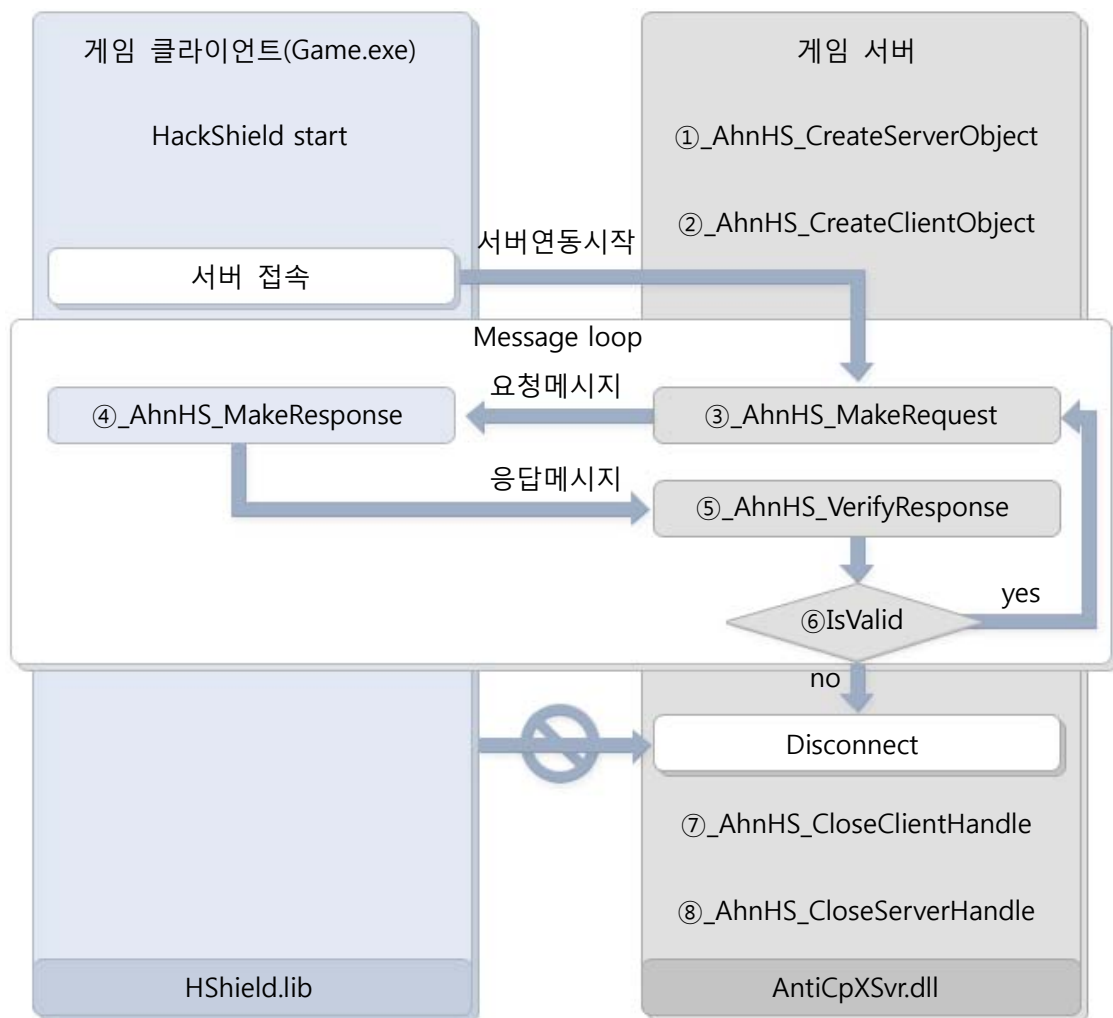


그림 1 HackShield 서버 연동 기능의 동작

위 동작 과정의 각 순서에 대한 자세한 설명은 다음과 같습니다.

- ①게임 서버가 최초 실행될 때 `_AhnHS_CreateServerObject()`를 통해 `AHNHS_SERVER_HANDLE`을 생성합니다. 이 핸들은 게임 서버가 종료되기 전까지 유지해야 합니다. 서버 핸들은 ②에서 클라이언트 핸들 생성 시 사용합니다.
- ②각 게임 클라이언트가 접속할 때 마다 `_AhnHS_CreateClientObject()`와 서버 핸들을 파라미터로 하여 `AHNHS_CLIENT_HANDLE`을 생성합니다. 이 핸들은 클라이언트 네트워크 접속이 유지하는 세션 동안 유지합니다. 게임 클라이언트 핸들은 ③의 요청메시지 생성 시 사용합니다.
- ③게임 클라이언트의 위/변조 여부 감시를 위해 요청메시지를 생성하여 전송합니다. 요청메시지는 `_AhnHS_MakeRequest()`를 사용하여 생성하며 클라이언트 핸들을 파라미터로 사용합니다.
- ④게임 클라이언트는 서버의 요청메시지에 적절한 응답메시지를 생성합니다. 응답메시지는 `_AhnHS_MakeResponse()`를 통해 생성합니다.
- ⑤게임 클라이언트의 응답 메시지가 유효한지 검사합니다. 응답 메시지의 유효성 검사는 `_AhnHS_VerifyResponse()`를 통해 검사합니다.
- ⑥ `_AhnHS_VerifyResponse()`에서 `ERROR_SUCCESS` 값 이외의 에러 코드가 발생했을 경우 에러 코드에 따라 적절하게 에러를 처리한 후 게임클라이언트 접속을 중단합니다.
- ⑦게임 클라이언트 접속을 끊을 때, 즉 세션이 종료될 때 ②에서 생성한 클라이언트 핸들을 닫습니다.
- ⑧게임 서버 프로세스가 종료될 때 ①에서 생성한 서버 핸들을 닫습니다. 이 때 클라이언트 핸들은 모두 닫힌 상태여야 합니다.

2장 적용

- 2.1 적용 준비 /12
- 2.2 서버 적용 /14
- 2.3 클라이언트 적용 /19
- 2.4 적용 시 주의사항 /21

2.1 적용 준비

HackShield 서버 연동 기능을 적용하기 전에 준비해야 하는 사항은 다음과 같습니다.

- 서버 관련 준비
- 클라이언트 관련 준비
- hsb 파일 작성

서버 관련 준비

게임 서버와 관련하여 준비해야 하는 사항은 다음과 같습니다.

- 라이브러리 추가
 - \Lib\Win\x86\AntiCpXSvr.lib 를 적당한 경로에 복사한 다음 게임 서버 프로젝트에 추가
- 헤더 파일 추가
 - \Include\AntiCpXSvr.h 를 적당한 경로에 복사
- 기타 파일 추가
 - \Bin\Win\x86\AntiCrack\AntiCpXSvr.dll 을 게임 서버의 실행 경로에 복사
 - \Bin\Win\x86\AntiCrack\HSPub.key 를 생성될 hsb 파일과 동일한 경로인 게임 서버의 실행 경로에 복사
 - \Bin\Win\x86\AntiCrack\3n.mhe 를 생성될 hsb 파일과 동일한 경로인 게임 서버의 실행 경로에 복사
 - \Bin\Win\x86\HShield\hshield.dat 를 생성될 hsb 파일과 동일한 경로인 게임 서버의 실행 경로에 복사

참고

3n.mhe 와 hshield.dat 파일 복사는 HackShield 버전 관리를 위해 권장합니다. 서버에서 hsb의 존재 경로에 게임 클라이언트에서 사용하는 3n.mhe 와 hshield.dat를 복사해 넣으면 3n.mhe 와 hshield.dat 버전이 올바르지 않은 게임 클라이언트를 접속하지 못하게 관리할 수 있습니다.

클라이언트 관련 준비

게임 클라이언트와 관련하여 준비해야 하는 사항은 다음과 같습니다.

- 라이브러리 추가
 - \Lib\Win\x86\HShield.lib 를 적당한 경로에 복사한 다음 게임 클라이언트 프로젝트에 추가
- 헤더 파일 추가
 - \Include\HShield.h 를 적당한 경로에 복사

hsb 파일 작성

WBin\Win\Wx86\AntiCrack\HSBGen.exe를 이용하여 hsb 파일을 생성합니다.

! 주의

클라이언트 실행 파일은 패킹되기 이전의 원본 릴리즈 파일이어야 합니다.

! 주의

디지털 서명된 클라이언트 실행 파일을 배포 시 HSBGen 툴 사용 후 게임 클라이언트 실행 파일에 디지털 서명 작업하십시오.

HSBGen.exe를 통해 hsb 파일을 생성하는 방법은 다음과 같습니다.

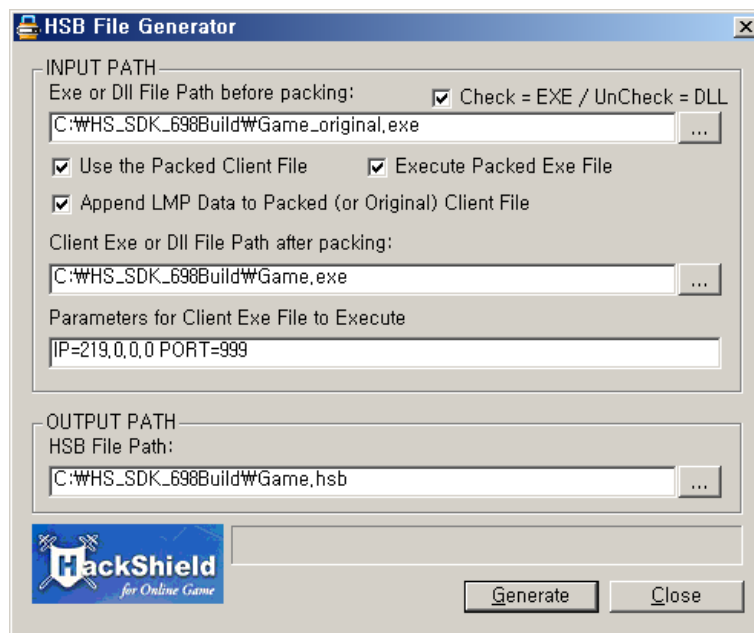


그림 2 HSBGen.exe로 hsb 파일 생성

1. 대상 파일이 exe 이면 **Check**, DLL 이면 **UnCheck** 를 합니다.
2. **Exe or Dll File Path before packing** 입력 상자에 패킹하지 않은 원본 실행 파일의 경로를 설정합니다. ...을 눌러 파일을 선택합니다.
3. 패킹된 클라이언트 실행 파일을 배포할 경우, 앞서 설명한 첫 번째 필드에 패킹되기 이전의 파일 경로를 입력합니다.
4. **Use the Packed Client File** 항목을 선택하고 **Client Exe or Dll File Path after packing** 입력 상자에 패킹된 파일 경로를 입력합니다.
5. **Execute Packed Client File** 은 **Client Executable File Path after packing** 에서 입력한 실행 파일을 자동으로 실행한 후 hsb 파일을 생성할 것인지를 결정합니다.

! 주의

패킹 프로그램이 upx 이외의 다른 패커(themida, asprotect 등..)로 패킹된 경우 execute client file 항목을 선택하십시오.

6. **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.
7. **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.
8. **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.
9. 모든 설정이 끝나면 **Generate** 를 누릅니다. 진행 상황이 프로그레스 막대에 표시되며 완료와 함께 정상적으로 완료했다는 메시지가 나타납니다.
10. .hsb 파일 생성이 완료되면 서버에서 로딩할 수 있는 특정 경로에 복사합니다.

2.2 서버 적용

서버에 적용하는 순서는 다음과 같습니다.

1. 헤더 파일 선언
2. 초기화
3. 클라이언트 핸들 생성
4. 요청메시지 생성 처리
5. 응답메시지 검증
6. 클라이언트 핸들 종료
7. 서버 핸들 종료

서버 적용 순서에 대한 자세한 설명은 다음과 같습니다.

헤더 파일 선언

서버 프로젝트에 앞서 복사한 AntiCpXSvr.h 파일을 경로에 맞게 선언합니다.

```
#include "AntiCpXSvr.h"
```

그림 3 헤더 파일 선언 예제

초기화

서버를 실행하기 전에 AntiCpXSvr 초기화하기 위해 _AhnHS_CreateServerObject()를 호출합니다.

```
// ① 게임 서버가 초기화 되는 부분에 _AhnHS_CreateServerObject를 호출합니다.
void GameServer_OnInitialize()
{
```

```

    ....
    AHNHS_SERVER_HANDLE hServer = NULL;

    // hsb 파일 경로
    hServer = _AhnHS_CreateServerObject ( g_szHsbFilePath );

    if ( hServer == ANTICPX_INVALID_HANDLE_VALUE )
    {
        printf ( "ERROR: _AhnHS_CreateServerObject\n" );
        return;
    }
    ....
}

```

그림 4 _AhnHS_CreateServerObject() 호출 예제

g_szHsbFilePath는 HSBGen.exe를 통해 생성한 hsb 파일의 전체 경로입니다.

클라이언트 핸들 생성

클라이언트가 세션을 생성할 때마다 클라이언트 핸들을 생성하기 위해 _AhnHS_CreateClientObject()를 호출합니다.

```

void GameServer_OnLogOn()
{
    ....

    AHNHS_CLIENT_HANDLE hClient = ANTICPX_INVALID_HANDLE_VALUE;
    unsigned long ulRet = ERROR_SUCCESS;
    AHNHS_TRANS_BUFFER stRequestBuf;

    // 해당 서버의 핸들을 이용해 접속한 유저의 핸들을 생성
    hClient = _AhnHS_CreateClientObject ( hServer );

    assert ( hClient != ANTICPX_INVALID_HANDLE_VALUE );
    if ( hClient == NULL )
    {
        printf ( "ERROR: _AhnHS_CreateClientObject\n" );
        return;
    }
}

```

그림 5 _AhnHS_CreateClientObject() 호출 예제

요청메시지 생성 처리

클라이언트와의 통신을 처리하는 스레드 함수 안에 요청메시지 생성 처리 모듈을 추가하기 위해 `_AhnHS_MakeRequest()`를 호출합니다.

```
// 해당 유저의 핸들을 이용해 요청메시지 작성
// 각 유저의 요청메시지는 해당 유저의 핸들을 이용해서 작성
ulRet = _AhnHS_MakeRequest ( hClient, &stRequestBuf );

if ( ulRet != ERROR_SUCCESS )
{
    printf ( "ERROR: _AhnHS_MakeRequest() == %Xh\n", ulRet );
    break;
}

// Socket을 통해 stRequestBuf 전송

....

send ( sock, stRequestBuf.byBuffer, stRequestBuf.nLength, 0 );
```

그림 6 `_AhnHS_MakeRequest()` 호출 예제

서버에서 사용하는 `_AhnHS_MakeRequest()`와 클라이언트에서 사용하는 `_AhnHS_MakeResponse()`에서는 `AHNHS_TRANS_BUFFER` 구조체를 버퍼로 입력받아 결과를 출력합니다. 버퍼의 구조는 다음과 같습니다.

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

그림 7 `AHNHS_TRANS_BUFFER` 구조체

실제 생성된 메시지는 배열 `byBuffer`에 저장되며 해당하는 배열의 길이는 `nLength`에 출력됩니다.

응답메시지 검증1

클라이언트로부터 요청메시지에 대한 응답메시지가 전송되었을 때 이를 검증하는 모듈을 추가하기 위해 `_AhnHS_VerifyResponse()`를 호출합니다.

```
void GameServer_VerifyReponse()
{
    AHNHS_TRANS_BUFFER stResponseBuf; // 클라이언트로부터 수신한 응답 버퍼

    ....

    // Socket을 통해 stResponseBuf 수신
```



```

.....

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
    )
{
    printf ( "ERROR: _AhnHS_VerifyResponse() = %X\n", ulRet );
}

printf( "SUCCESS: hClient = %d\n", hClient );

.....
}

```

그림 8 _AhnHS_VerifyResponse() 호출 예제

서버가 _AhnHS_VerifyResponse()를 호출하여 클라이언트로부터 받은 버전 응답메시지를 분석합니다. 만약 위 예제의 if문에서 처리한 메시지가 아니라면 정상적인 경우이므로 계속 진행합니다. 그렇지 않은 경우 에러 처리를 한 후 해당 클라이언트의 소켓을 종료하고 접속 리스트에서 제거합니다.

응답메시지 검증2

클라이언트로부터 요청메시지에 대한 응답메시지가 전송되었을 때 이를 검증하는 함수인 _AhnHS_VerifyResponse() 대신 _AhnHS_VerifyResponseEx()를 호출합니다.

```

void GameServer_VerifyReponse()
{

```

```

        AHNHS_TRANS_BUFFER stResponseBuf; // 클라이언트로부터 수신한 응답 버퍼
        ULONG ulLastError = 0;
        ....

        // Socket을 통해 stResponseBuf 수신

        ....

        ulRet = _AhnHS_VerifyResponseEx (    hClient,
                                            stResponseBuf.byBuffer,
                                            stResponseBuf.nLength,
                                            &ulLastError);

        if ( ulRet == ANTICPX_RECOMMAND_CLOSE_SESSION )
        {
            //게임 클라이언트에서 해킹 행위가 감지되었으므로,
            //게임 클라이언트와의 접속을 차단하도록 합니다.
        }

        printf( "SUCCESS: hClient = %d\n", hClient );

        ....
    }

```

그림 9 _AhnHS_VerifyResponseEx() 호출 예제

위 예제에 대한 설명은 다음과 같습니다.

1. 서버는 _AhnHS_VerifyResponseEx()를 호출하여 클라이언트로부터 받은 버전 응답메시지를 분석합니다.
2. 함수의 반환값이 ANTICPX_RECOMMAND_KEEP_SESSION 이면 정상이므로 계속 진행합니다. 만약 ANTICPX_RECOMMAND_CLOSE_SESSION 이면 에러 처리한 후 해당 클라이언트의 소켓을 종료하고 접속 리스트에서 제거합니다.
3. _AhnHS_VerifyResponseEx()는 _AhnHS_VerifyResponse() 호출 시 반환값이 다음 값들 중 하나인 경우 ANTICPX_RECOMMAND_CLOSE_SESSION 을 리턴합니다.

```

- 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_ABNORMAL_HACKSHIELD_XSTATUS
- ERROR_ANTICPXSVR_OLD_HACKSHIELD_VERSION
- ERROR_ANTICPXSVR_DETECT_CALLBACK_IS_NOTIFIED

그림 10 ANTICPX_RECOMMAND_CLOSE_SESSION 반환값

클라이언트 핸들 종료

클라이언트가 접속을 중지하여 세션이 종료되면 _AhnHS_CloseClientHandle()을 호출하여 클라이언트 핸들을 닫습니다.

```
void GameClient_OnFinalize()
{
    ....
    _AhnHS_CloseClientHandle ( hClient );
    ....
}
```

그림 11 _AhnHS_CloseClientHandle() 호출 예제

서버 핸들 종료

게임 서버를 종료할 때 _AhnHS_CloseServerHandle()을 호출하여 서버 핸들을 닫습니다.

```
void GameServer_OnFinalize()
{
    ....
    _AhnHS_CloseServerHandle ( hServer );
    ....
}
```

그림 12 _AhnHS_CloseServerHandle() 호출 예제

2.3 클라이언트 적용

클라이언트에 적용하기 위해 응답메시지를 처리합니다. 클라이언트 적용 순서에 대한 자세한 설명은 다음과 같습니다.

응답메시지 처리

서버와의 통신을 처리하는 스레드 함수 안에 요청메시지에 대한 응답메시지 처리 모듈을 추가하기 위해 _AhnHS_MakeRequest()를 호출합니다.

```
void GameClient_MakeResponse()
{
    AHNHS_TRANS_BUFFER stRequestBuf; // 서버로부터 수신한 요청메시지
```

```

        AHNHS_TRANS_BUFFER stResponseBuf;    // 서버로 전송할 응답메시지

        ....

        // Socket을 통해 stRequestBuf 수신

        ....

        ulRet = _AhnHS_MakeResponse ( stRequestBuf.byBuffer,
                                      stRequestBuf.nLength,
        &stResponseBuf );

        if ( ulRet != ERROR_SUCCESS )
        {
            printf ( "ERROR: _AhnHS_MakeResponse() == %Xh\n", ulRet );
        }

        // Socket를 통해 stResponseBuf 서버로 전송

        ....

        send ( sock, stResponseBuf.byBuffer, stResponseBuf.nLength, 0 );
    }

```

그림 13 _AhnHS_MakeRequest() 호출 예제

서버로부터 요청메시지를 수신하면 그에 해당하는 응답메시지를 작성합니다. 클라이언트의 메시지 처리 루틴에 요청메시지 수신 부분을 추가하고 _AhnHS_MakeResponse()를 호출합니다.

2.4 적용 시 주의사항

적용 시 주의해야 할 사항은 다음과 같습니다.

에러 메시지 출력

디버깅 및 고객 지원을 위해 에러 메시지를 출력할 때 에러 코드를 함께 출력하는 것이 좋습니다.

에러 발생 시 처리 방법

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.

서버 요청에 응답이 없는 경우

서버 요청에 응답이 없는 경우 커넥션을 종료하도록 구성합니다. 권장 응답 시간은 10초입니다.

HSB 파일 생성 시

HSBGen.exe로 hsb 파일을 생성할 때 클라이언트 실행 파일은 패킹되기 이전의 원본 파일이어야 합니다.

Linux/Unix 환경에서 대소문자 구분

Linux/Unix 환경에서는 대소문자를 구분하므로 HSPub.key 파일 이름을 정확하게 입력합니다.

3장

테스트 및 배포

3.1 테스트 /23

3.2 배포 /23

3.1 테스트

작성한 소스 코드를 테스트합니다. 클라이언트의 실행 파일을 에디터로 열어 특정 부분을 변조한 다음 실행시켜서 서버에 접속했을 때 연결이 끊기는 것을 확인합니다. 이 테스트를 통해 서버 연동 동작 여부를 판단할 수 있습니다.

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.

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.