AhnLab HackShield Pro for Online Game

Quick Guide

V 2.0

실행 파일 실시간 복호화 기능

Ahn 안철수연구소

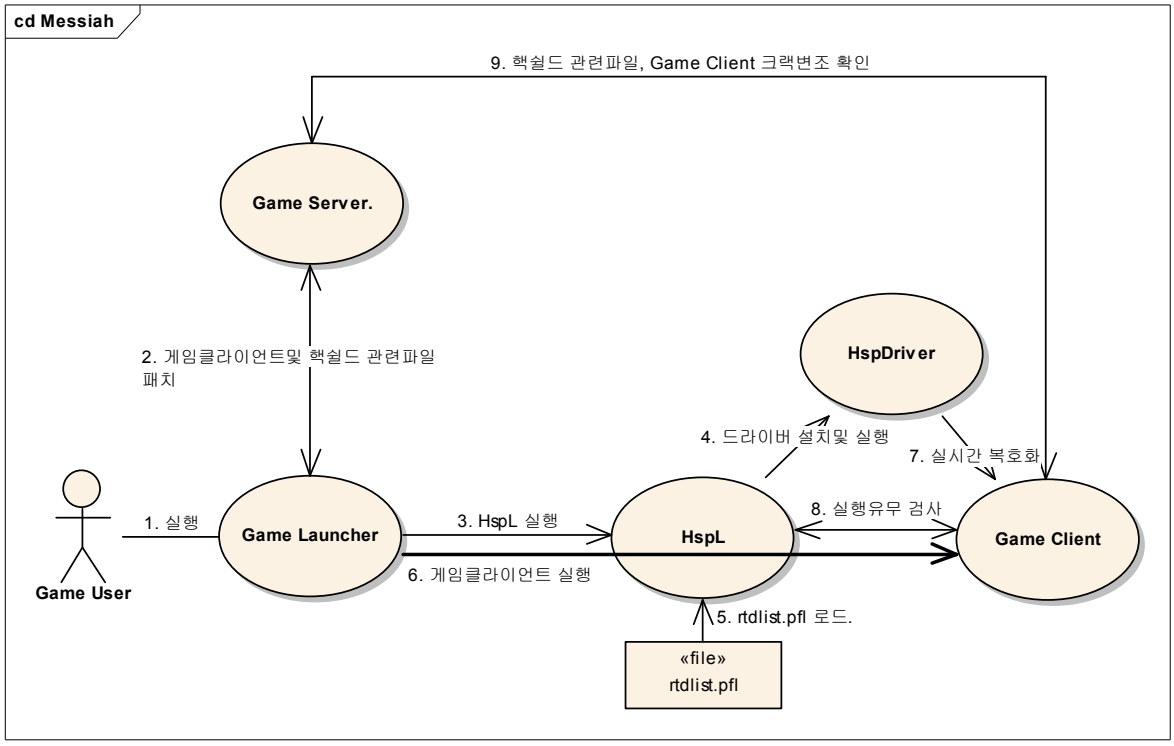
1 시작하기 전에

- ❖ 이 문서는 HackShield Pro(이하 핵쉴드) SDK 적용을 위한 가이드라인입니다. Microsoft Visual C++ 6.0 환경에서 C/C++ 언어로 작성하는 것을 기준으로 작성되었습니다. 핵쉴드의 실시간 복호화 기능에 대한 부분을 다룹니다.
- ❖ 처음 핵쉴드 코드 적용을 돕기 위하여 대략적인 설명과 간략한 샘플 코드를 다룹니다. 세부 구현은 각 게임 개발사의 정책에 맞추어 진행하십시오.
- ❖ \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 실시간 복호화 기능의 구성 파일

- ❖ \Bin\HShieldPro\HCryptor.exe : 실행 파일 암호화 툴
- ❖ \Bin\HShieldPro\HspL.exe : HackShield Pro Launcher 툴

3 동작 구조



① 게임 사용자가 게임 런처를 실행합니다.

② 게임 런처는 게임 서버에게 최신 버전의 게임 클라이언트 파일, HspL, HackShield 파일을 요구하여 다운로드 받습니다.

③ 다운로드를 완료한 다음 HspL를 실행시키면 HspL는 HackShield를 실행시켜 게임을 보호합니다.

④ HackShield가 정상적으로 실행되면 HspDriver를 실행시킵니다.
⑤ 게임 클라이언트를 실행하기 전에 실시간 복호화에 필요한 정보를 rtdlist.pfl 파일에서 로드합니다. rtdlist.pfl파일은 HspCryptor가 중요한 파일을 암호화할 때 생성되는 파일이며 HspL와 같은 위치에 존재해야 합니다.
⑥ HspL를 통하여 게임 클라이언트를 실행시킵니다.
⑦ 이 때 HspDriver는 게임 클라이언트를 실시간으로 복호화합니다.
⑧ HspL은 주기적으로 자신이 생성한 자식 프로세스(게임 클라이언트)의 존재 유무를 검사합니다.
⑨ 게임 서버는 주기적으로 게임 클라이언트와 HackShield 관련 모듈의 파일 변조 여부를 검사하여 크랙 여부를 확인합니다.

4 적용 전 준비사항

❖ \Bin\HShieldPro\HCryptor.exe를 이용하여 암호화된 게임의 실행 파일과 rtdlist.pfl 파일을 생성합니다.
  ✓ HCryptor 의 사용법은 "*AhnLab HackShield 프로그래밍 가이드: 8.2. HackShield Cryptor 툴 사용 방법*"을 참고하십시오.
  ✓ 생성된 암호화된 실행 파일과 rtdlist.pfl 파일은 게임 실행 경로에 복사합니다.
❖ 암호화된 게임의 실행 파일의 실시간 복호화를 위하여 일반적으로 게임 런처를 작성합니다.
❖ \Lib\HspLLib.lib 라이브러리 파일이 필요합니다. 적당한 위치에 복사한 후 게임 런처 프로젝트에 추가합니다.
❖ \Include\HspLauncherLib.h 헤더 파일이 필요합니다. 적당한 위치에 복사합니다.
❖ \Bin\HShieldPro\HspL.exe 파일이 필요합니다. 게임의 실행 경로에 복사합니다.

5 게임 런처 작성

❖ 게임 런처 프로젝트에 앞서 복사한 HspLauncherLib.h 파일을 위치에 맞게 인클루드시킵니다.

```
#include "HspLauncherLib.h"
```

❖ HspL를 실행합니다. (_AhnHsp_StartLauncher)

```
DWORD dwRet = ERROR_SUCCESS;

dwRet = _AhnHsp_StartLauncher(m_strLauncherPath);

if( dwRe t != ERROR_SUCCESS )
{
    // 에러 처리
}
```

  ✓ _AhnHsp_StartLauncher 함수 호출이 성공적으로 완료되어야 이후의 실행 파일 실시간 복호화를 할 수 있습니다.
  ✓ _AhnHsp_StartLauncher의 인자인 m_strLauncherPath는 실행할 HspL의 전체 경로이며 HspL은 게임이 실행되는 폴더에 위치해야 합니다.
  ✓ 에러 처리에 대한 자세한 사항은 "*AhnLab HackShield 프로그래밍 가이드: 4.3.*

Application Programming Interface"를 참고하십시오.

❖ HspL를 통해 암호화된 게임 실행 파일을 실행합니다. (_AhnHsp_StartGame)

```
// ① AHN_GAMEEXECINFO는 게임 실행과 관련한 구조체 입니다.
AHN_GAMEEXECINFO    GameExecInfo;

// ② 구조체에 해당하는 값을 채워 줍니다.
sprintf( GameExecInfo.szFilePath, "%s", m_strTargetPath );
sprintf( GameExecInfo.szCommandLine, "%s", m_strOption);

// ③ _AhnHsp_StartGame 함수를 호출하여 HspL를 통해 암호화된 실행 파일을 실행합니다.
dwRet = _AhnHsp_StartGame( &GameExecInfo );

// ④ _AhnHsp_StartGame 함수의 리턴 값을 검사하여 에러 처리합니다.
if( dwRet != ERROR_SUCCESS )
{
    // 에러 처리
}
```

① _AhnHsp_StartGame 함수 호출에는 AHN_GAMEEXECINFO 구조체가 필요합니다. 이 구조체는 게임 실행 파일의 전체 경로와 게임 실행에 필요한 옵션 정보로 구성됩니다.

② AHN_GAMEEXECINFO 구조체에 값을 채웁니다. szFilePath는 실행 파일의 이름을 포함한 실행 파일의 전체 경로입니다(ex. C:\Program Files\Game\Game.exe). szCommandLine에는 게임 실행 시 필요한 인자 값을 넣어줍니다(ex. 게임을 실행할 때 Game.exe abc 로 실행한다면 인자 값 abc를 넣어줍니다).

③ _AhnHsp_StartGame 함수 호출에는 앞서 값을 채운 구조체 AHN_GAMEEXECINFO가 인자로 필요합니다. 함수가 호출되면 암호화된 게임의 실행 파일을 실행합니다.

④ _AhnHsp_StartGame 함수의 리턴 값을 받아서 정상 수행(ERROR_SUCCESS)이 아니라면 해당하는 에러 코드에 따라 에러 메시지를 출력하고 종료합니다. 각 경우에 대한 에러 메시지는 "*AhnLab HackShield 프로그래밍 가이드: 4.3. Application Programming Interface*"를 참고하십시오.

❖ 게임 런처를 종료할 때 HspL를 해제합니다.

```
_AhnHsp_CloseHandle();
```

6 테스트 및 배포

❖ 작성된 게임 런처를 게임의 실행 경로에 복사합니다. (최종적으로 게임의 실행 경로에 암호화된 게임 실행 파일, HspL.exe, rtdlist.pfl, 게임 런처가 존재해야 합니다)

❖ 암호화된 게임 실행 파일을 실행하여 실행되지 않는 것을 확인합니다.

❖ 게임 런처를 실행하여 게임이 정상적으로 실행되는지 확인합니다.