nProtect GameGuard CSAuth30 Manual
-Complete Document-

■ (주)잉카인터넷 게임보안 사업본부 글로벌사업부

SE1-130228

Product Ver. 2.15

계약납품용

INCA
I.n.t.e.r.n.e.t

nProtect GameGuard Pro

INDEX

1장 CS인증

1. CS인증의 소개

① CS 인증 이란?

게임 서버는 온라인 게임에서 가장 중요한 요소이며 해킹에 대한 강력한 보안과 일괄적인 시스템 구성으로 안정성이 보장되어 있습니다. 그에 비해서 보안에 취약한 게임 클라이언트는 언제든지 해킹의 위협에 노출되어 있으며 해커는 게임서버로의 접근이 유일하게 보장된 게임 클라이언트를 통해서 게임 해킹을 시도합니다. 그러므로 게임서버는 항상 게임 클라이언트의 정상 동작을 검사해야 하며 부정 사용자에 대한 적발 및 제재방법이 필요하게 됩니다.

㈜잉카인터넷은 온라인게임에서의 악성코드에 대한 진단 및 차단 서비스를 다년간 제공하여 왔으며, 고객의 여러 요구사항을 수용하였습니다. 게임 보안 기술의 KNOW-HOW를 바탕으로 nProtect GameGuard는 게임서버와 게임클라이언트의 보다 더 완벽한 보안을 위해 CS인증 시스템을 개발하였습니다.

nProtect GameGuard는 클라이언트 측면에서 게임핵 프로그램 사용이나, 해킹의 시도 등을 거의 완벽하게 방어할 수 있는 강력한 게임보안용 프로그램입니다. GameGuard 에 의해서 보호된 게임클라이언트는 보고된 모든 핵툴에 대한 해커의 클라이언트의 변조 방지 및 비정상 동작 방지가 보장하고 있으며 신속한 GameGuard 의 업데이트를 통해서 최신의 해킹 방지 기술을 적용함으로써 보다 신뢰성 있는 서비스를 제공하고 있습니다.

하지만 게임클라이언트의 보안은 GameGuard 가 존재해야만 보장 받을 수 있습니다. CS인증은 최신의 GameGuard가 정상적으로 동작하고 있는지 서버에서 확인함으로써 비정상적인 서버 접속 및 변조된 게임클라이언트의 접근을 원천적으로 차단할 수 있습니다.

② CS 인증이 필요한 이유

- GameGuard 의 동작을 인위적으로 중단시키고 변조된 게임클라이언트가 서버에 접속하는 경우
- 게임클라이언트의 변조로 GameGuard 를 실행시키지 않는 경우
- 게임클라이언트를 실행시키지 않고 해킹 프로그램으로 직접 서버에 접속하는 경우
- 최신 게임가드의 해킹 검출 기법을 회피하기 위해 구 버전 GameGuard 과 해킹 프로그램을 같이 동작시키는 경우

위 4가지 취약점의 대응 방법은 온라인 게임 요소 중에 유일하게 신뢰할 수 있는 요소인 게임 서버에서 GameGuard 의 정상 동작을 검사하는 방법 밖에 없으며 CS인증은 항상 최신 GameGuard 의 정상동작을 게임서버에서 검사함으로써 보다 향상된 게임클라이언트 보안서비스를 제공합니다.

더욱이 CS인증 3.0 에서는 서버를 활용한 보안 기능을 사용할 수 있으므로 핵툴 억제력이 크게 향상 됩니다.

③ CS 인증의 특징

- CS인증 알고리즘의 보호
  CS인증 알고리즘은 GameGuard 의 코드 내부에 포함되어 있으므로 분석이 매우 어렵습니다.

- 인증 알고리즘의 이중 암호화
  GameGuard 의 코드영역에서 CS인증 알고리즘 부분만 한번 더 암호화 하여 서버에서 받은 키로 암호화된 코드를 풀지 않는 한 알고리즘 분석이 불가능 하므로 더욱더 보안성이 강화되었습니다.

- 인증패킷의 개별성
  모든 인증 패킷은 인증 요청을 할 때마다 난수와 key 의 조합으로 구성하므로 각각의 인증패킷은 항상 유일함을 보장합니다.

- 인증 계산 알고리즘의 실시간 변경
  인증 계산 알고리즘은 인증 모듈 별로 실시간으로 계산알고리즘이 변경되어서 하나의 계산 알고리즘이 분석되더라도 바로 다른 계산 알고리즘으로 변경되어 개별적인 분석으로는 인증을 회피하기 힘듭니다.

- 최신 버전의 GameGuard 보장
  서버에서는 항상 최신 버전의 GameGuard 의 접속만을 허용함으로 새로운 해킹 방법이 배포되더라도 GameGuard 의 버전을 올려서 업데이트 함으로써 즉각적인 대응이 가능하며 구 버전 인증으로의 접속을 제한하게 됩니다.

- GameGaurd 정상 동작 여부 확인
  GameGuard 가 비정상 동작하던가 멈춰 있는 것을 감지하여 알고리즘만을 사용하는 서버인증 우회 방식에 대응 합니다.

- PC 유일값을 이용하여 논클라이언트 차단
  한대의 PC 에서 여러 개의 계정을 이용하는 것을 감지하여 논클라이언트 사용을 차단합니다.

- 유저별 Key 를 생성하여 핵툴 서버 차단
  유저별로 다른 Key 값을 사용함으로 한 개의 GameGaurd 에서는 한 유저의 인증만 할 수 있도록 하여 핵툴 서버를 차단합니다.

- GameGuard 바이너리 파일 복잡화
  내부버전 변경 시마다 GameGuard 모듈의 바이너리를 뒤섞음으로 분석을 어렵게 만들어 모듈의 수명, 핵툴의 업데이트 주기를 늘립니다.

④ CS 인증 3.0 의 변경/추가된 기능

1. 기능 확장성 확보

- 기존의 16바이트 구조로는 기능의 추가가 불가능 함으로서
  패킷의 크기를 16바이트에서 최대 4096 바이트의 가변사이즈로 변경 하였습니다.
  (서버인증 2.6 에서는 비트 단위로 기능을 삽입하여 포화 상태가 됨)
  패킷의 크기가 커짐으로서 서버를 통한 보안 기능을 쉽게 추가할 수 있도록 하였습니다.

2. 유지 보수성 증대

- 기존의 서버인증 2.5 / 2.6 은 중요 기능이 라이브러리에 포함되어 있으므로
  문제가 발생시에 서버를 재 빌드 해야만 하는 불편함이 있었습니다.
  3.0 은 모든 기능을 .dll/.so 파일에 구현함으로서 단순히 파일 교체만으로
  업데이트가 가능 하도록 하였습니다.
- 모든 코드에 OOP 를 적용하여 내부적으로 유지보수가 쉽도록 변경 하였습니다.
- 에러코드를 세분화 하여 문제 발생시 쉽게 대응할 수 있도록 하였습니다.
- 로그를 직접 남김으로서 문제 발생시 로그를 통한 확인이 쉽도록 처리 하였습니다.
- 이외에 게임가드 측에서 쉽게 유지보수 하기 위한 다양한 기능이 추가 되었습니다.

3. 사용성 증대

- 서버인증의 옵션을 .ini 파일로 분리함으로서 서버의 재빌드 / 모듈 교체 없이
  설정을 변경 할 수 있도록 하였습니다.

4. 업체 적용의 편의성

- 함수를 최대한 줄이고 자동화가 가능한 부분은 모두 자동화 하였습니다.
- 함수의 인터페이스를 최대한 간편화 하였습니다.
- 메뉴얼을 따르지 않은 다양한 방식의 적용에도 최대한 문제가 생기지 않도록 처리 하였습니다.

5. 정보 저장 가능

- 서버가 다시 시작 되더라도 필요한 정보를 미리 파일에 저장함으로서 정보의 연속성을 가지도록 하였습니다.

6. 신기능 추가

- Scan 패턴이 서버를 통하는 기능을 적용하여 패턴 추가 없이 새로운 핵툴에 대해 자동으로 차단되는 기능을 추가하였습니다.
- 이외에 몇가지 다양한 핵툴 감지 기능이 추가되었습니다.

7. 메뉴얼 업데이트

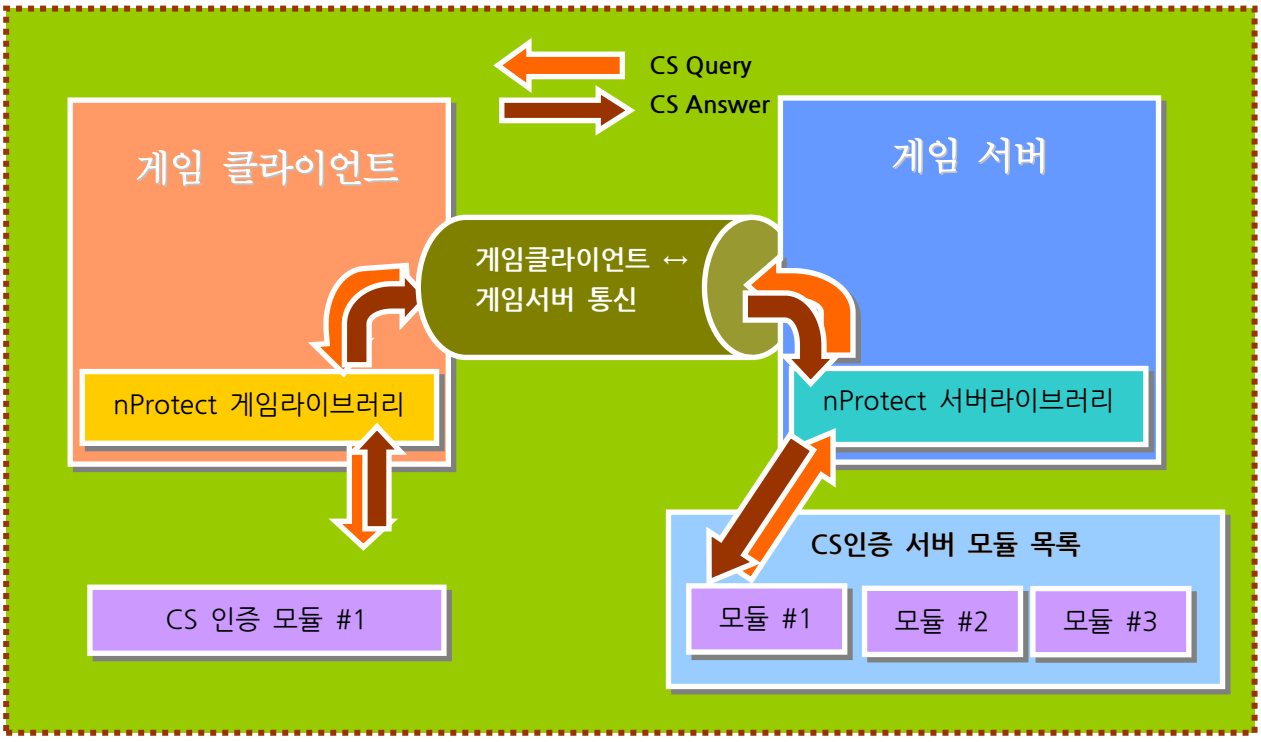
- 최근에 이슈가 되었던 문제들을 메뉴얼에 기본적으로 추가 함으로서 게임사에서 메뉴얼만 숙지한다면 모든 서버인증 문제에 대해서 대처할 수 있도록 하였습니다.

8. 구버전 게임 클라이언트 차단

- 구버전 게임 클라이언트를 차단하는 기능이 추가 되었습니다. 서버에 지속적으로 최신 파일을 등록해주는 수동방식과 현재 접속자의 통계를 이용하여 최신버전을 판단하는 자동 방식을 제공합니다.

2. 시스템 구성

① 시스템 구성도



② 구성 모듈

- 서버에 적용할 파일들로만 구성됨. 클라이언트는 NPGameLib.lib 에 의해 제공됨
- 32 는 32bit 서버용 모듈, 64 는 64bit 서버용 모듈을 의미
- ggauth32_x.dll(so), ggauth64_x.dll(so) 파일은 이름 변경 하지 말 것

<기본>

설정파일 : csauth3.cfg

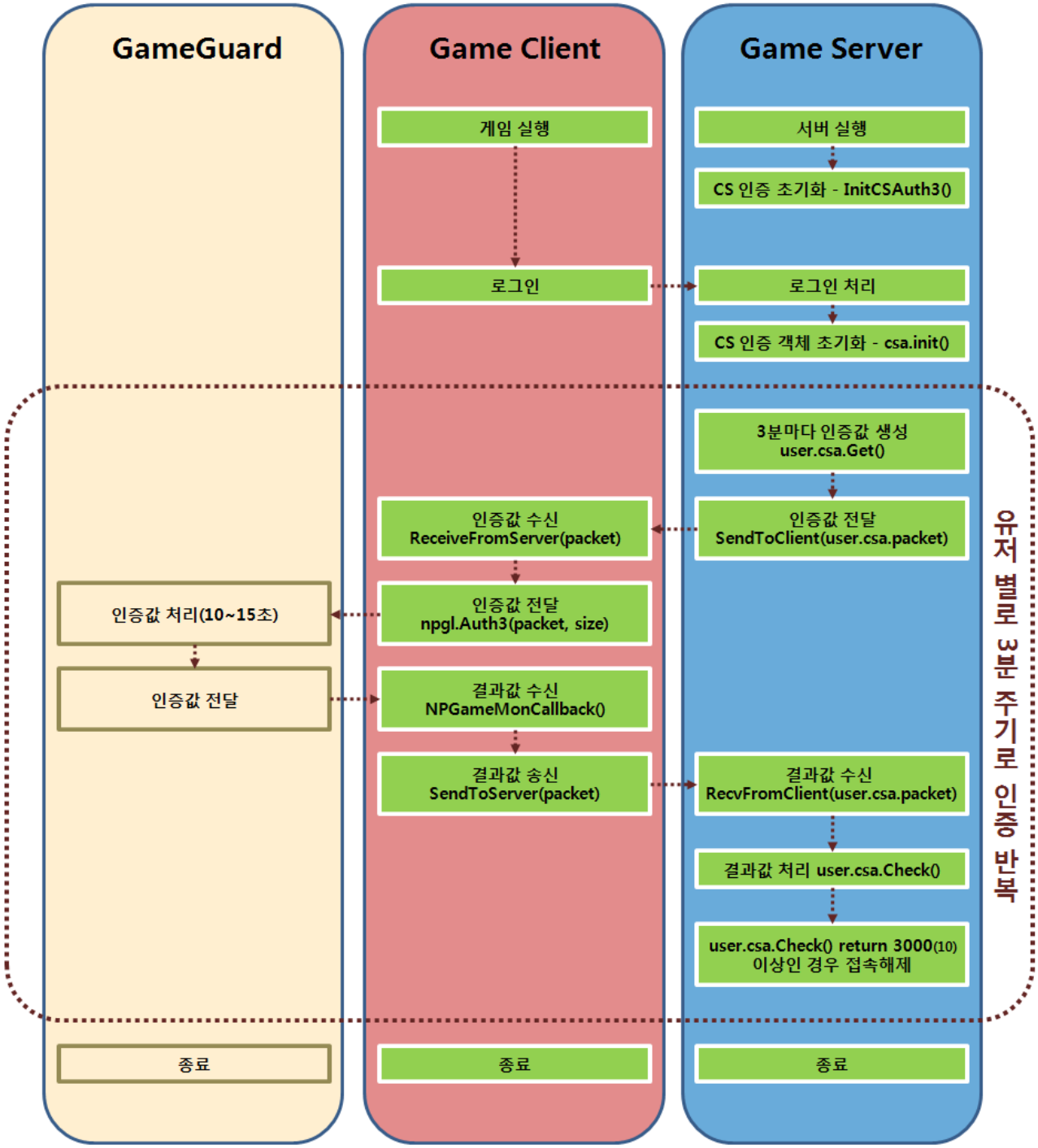
<Windows>

- c/c++ 사용시
  - 헤더 : ggsrv30.h
  - 라이브러리 : ggsrv30lib,lib
  - 모듈 : ggauth32_x.dll
- 이외의 언어 사용시 (Delphi 등)
  - 헤더 : ggsrv30.h **(#define _HEADER_DLL 주석해제)**
  - 라이브러리 : ggsrv30,dll
  - 모듈 : ggauth32_x.dll

<Linux>

- dynamic build 사용시
  - 헤더 : ggsrv30.h
  - 라이브러리 : libggsrv30.a
  - 모듈 : ggauth32_x.so
- static build 사용시
  - 헤더 : ggsrv30.h
  - 라이브러리 : libggsrv30_x_static.a **(ggauth32_x.so 미사용)**
- c 컴파일러 사용시
  - 헤더 : ggsrv30.h **(#define _HEADER_C 주석 해제)**
  - 이하 위와 동일

③ 프로세스


GameGuard
Game Client
Game Server
게임 실행
서버 실행
CS 인증 초기화 - InitCSAuth3()
로그인
로그인 처리
CS 인증 객체 초기화 - csa.init()
3분마다 인증값 생성
user.csa.Get()
인증값 수신
ReceiveFromServer(packet)
인증값 전달
SendToClient(user.csa.packet)
인증값 처리(10~15초)
인증값 전달
npgl.Auth3(packet, size)
인증값 전달
결과값 수신
NPGameMonCallback()
결과값 송신
SendToServer(packet)
결과값 수신
RecvFromClient(user.csa.packet)
결과값 처리 user.csa.Check()
user.csa.Check() return 3000(10)
이상인 경우 접속해제
유저 별로 3분 주기로 인증 반복
종료
종료
종료

3. 시스템 적용

① 준비 사항

- 2-2. 구성모듈에서 필요한 모듈을 준비합니다.
- 헤더와 라이브러리 파일을 서버 프로젝트에 추가합니다.
- 모듈 파일(dll, so) 과 설정파일(csauth3.cfg) 은 서버내의 적당한 위치에 게임가드 전용 폴더(예: ./GameGuard) 를 생성하고 그 곳에 복사합니다.
  (두 파일은 항상 같은 폴더에 위치해야 합니다.)
  (주의 : ggauth32_xx 모듈파일은 뒤의 숫자를 지우거나 변경하면 안됩니다. InitCSAuth3 함수가 폴더를 검색하여 가장 최신의 모듈을 Load 하도록 구현되어 있습니다.)

② 서버 적용 (필수 구현)

가. CS인증 관련 함수를 사용하는 모든 소스파일에서 함수의 선언을 참조할 수 있도록 ggsrv30.h 를 include 를 하십시오.

```
#include "/GameGuard/ggsrv30.h"
```

나. 게임서버의 초기화가 완료된 후에 CS인증을 초기화 시키십시오.

```
        ...
        // 게임서버초기화완료

        // "./GameGuard" 는csauth3.cfg 파일과 ggauth3_x.dll(모듈) 이 있는 폴더
        // 끝에 slash (/ or ₩₩) 유무 상관없음
        UINT32 uReturn = InitCSAuth3("./GameGuard/");
        if( uReturn != ERROR_SUCCESS )
        {
                // error
                exit(1);
        }
```

다. 접속유저를 구분하는 구조체나 객체에 CCSAuth3 객체를 추가하고 사용전에 Init()을 호출합니다.
(CCSAuth3 객체는 접속 유저 수만큼 존재해야 합니다. 유저별 정보를 저장합니다.)
(참고3 : 서버이동시 주의점과 dwServerNumber 사용방법)

```
class CUser
{
        ...
public:
        CCSAuth3 csa;
}

int main(void)
{
        CreateUser(&pUser);
        pUser->csa.Init( true ); // 일반 적으로 항상 true 를 호출 합니다.
}
```

```
// C 의경우- malloc 으로 서버인증내에서 직접 메모리를 할당하는 경우
struct _USER
{
        ...
        PBYTE pCSInstance;
}

int main(void)
{
        CreateUser(&pUser);
        // malloc 으로 유저별 인스턴스를 할당함
        pUser->pCSInstance  = CSAuth3_CreateInstance();
        CSAuth3_Init( pbUser->pCSInstance, TRUE );
}
```

```
// C 의경우- 서버의 메모리풀을 사용하여 유저별 인스턴스를 직접 할당하는 경우
struct _USER
{
        ...
        PBYTE pCSInstance;
}

int main(void)
{
        CreateUser(&pUser);

        UINT32 uInstanceSize = CSAuth3_GetInstanceSize();
        // 서버내의 메모리풀의 메모리를 직접 할당함.
        pUser->pCSInstance = GetMemoryPool( uInstanceSize );

        CSAuth3_SetInstance( pUser->pCSInstance );
        CSAuth3_Init( pbUser->pCSInstance, TRUE );
}
```

라. 일정한 주기마다 CS인증 쿼리를 생성하여 클라이언트에게 전송합니다. (권장 주기는 3~5분)
(참고1 : 권장 주기)
(참고2 : timeout 구현 금지)
(참고5 : 에러코드 종류와 의미)

```
        // 타이머를사용하여3~5분에1회씩호출
        // timeout 은구현하지않는것을권장함
        // ( 게임가드측에서테스트를위한CS인증off 가불가능해짐으로핵툴분석에
        //   어려움이있을수있습니다. 또한정확한에러통계가되지않습니다. )
        UINT32 uReturnedPacketSize = 0;
        UINT32 uReturn = pUser->csa.Get( &uReturnedPacketSize );
        If( uReturn >= 3000 )
        {
                // 유저접속해제
                return;
        }

        // uReturnedPacketSize 은최대
```

```
// WARNING 류의에러는uReturnedPacketSize 가0으로리턴됨
If( uReturnedPacketSize > 0 )
        sendToClient(pUser->csa.packet, uReturnedPacketSize)
```

```
// C 의 경우
UINT32 uReturnedPacketSize = 0;
UINT32 uReturn = CSAuth3_Get( pUser->pCSInstance, &uReturnedPacketSize );
If( uReturn >= 3000 )
{
        return;
}

If( uReturnedPacketSize > 0 )
{
        PBYTE pPacket = CSAuth3_GetPacket( pUser->pCSInstance );
        sendToClient(pPacket, uReturnedPacketSize)
}
```

마. CS인증 응답을 받은 경우 유저별 응답을 처리합니다.

```
// 유저별로 클라이언트에서 전달받은 NPGameMonCallback  에서의
// pCSAuth3->bPacket 을 packet 에 복사하고
// pCSAuth3->dwPacketSize 을 return  한다.
UINT32 uReceivedSize = recvFromClient (pUser->csa.packet);
UINT32 uReturn = pUser->csa.Check( uReceivedSize );
if( uReturn >= 3000 )
{
        // 유저접속해제
        return;
}
```

```
// C 의 경우
BYTE packet[4096] = {0,};
UINT32 uReceivedSize = recvFromClient(packet);
UINT32 uReturn = CSAuth3_Check( pUser->pCSInstance, packet, uReceivedSize );
if( uReturn >= 3000 )
{
        return;
}
```

바. 사용자의 접속이 해제되었을 경우 CSAuth3 객체의 Close 를 호출합니다.
- 사용자의 접속 해제시에 동작하여야 할 기능이 다수 포함되어 있습니다.
- 반드시 호출이 되도록 소멸자에서도 Close 를 호출하고 있고 Init 이후에 재차 init 이 호출 되는 경우 내부에서 Close 를 호출 하고 있습니다.

```
// 사용자 접속이 해제된 경우
pUser->csa.Close();
LogOut(pUser);
```

```
// C 의경우- CSAuth3_CreateInstance() 를 통해서 인스턴스를 할당 받은 경우
CSAuth3_Close( pUser->pCSInstance );
LogOut( pUser );

// pUser 메모리해제전에 CSInstance 메모리 해제
// 내부에서 free / Close / 소멸자를호출함
CSAuth3_Release( pUser->pCSInstance )
```

```
        // C 의경우- 서버내의 메모리풀을 사용하여 인스턴스를 할당한 경우
        CSAuth3_Close( pUser->pCSInstance );
        LogOut( pUser );

        // 서버내의 메모리풀의 메모리를 직접 해제함
        UINT32 uInstanceSize = CSAuth3_GetInstanceSize();
        ReleaseMemoryPool( pUser->pCSInstance, uInstanceSize );
```

사. 서버를 종료시킬 때에는 CloseCSAuth3() 함수를 반드시 호출 합니다.
(종료 시에 게임가드 전용폴더에 몇가지 정보를 기록합니다.
서버 종료 시에 반드시 호출 되도록 해주시기 바랍니다.)

```
        // 서버가 종료되는 경우
        CloseCSAuth3();
```

③ 서버 적용 (선택 구현)

가. CS인증 3.0 의 로그를 전달 받기 위해서 callback 함수를 등록합니다.
(csauth3.cfg 의 설정에 따라서 CS인증 자체적으로 로그를 남기므로 선택 사항입니다.)
(참고3 : 로그의 종류)

```
// LOG_DEBUG 의 경우는 최대 10240 byte 정도의 큰 로그가 남을 수 있으므로 Buffer Overflow 에
// 주의하여 구현해 주시기 바랍니다.
void __stdcall CS3LogCallback( int nLogType, char *szLog )
{
        // 로그가 발생 시 callback  함수로 로그를 전달합니다.
        // callback  함수가 등록되어 있지 않다면 호출되지 않습니다.

        // 게임가드 프로토콜 버전, 내부버전의업데이트가발생했을때호출됩니다. (호출빈도 낮음)
        if( nLogType == LOG_UPDATE )
                printf("this is update log : %s₩n", szLog);

        // 초기화상황, 일반적인로그, 에러발생등의로그가발생했을때호출됩니다. (호출빈도 보통)
        else if( nLogType == LOG_NORMAL )
                printf("this is normal log : %s₩n", szLog);

        // 모든 유저의 패킷 정보가 남게 됩니다. (호출빈도,양 매우 높음)
        else if( nLogType == LOG_DEBUG )
                printf("this is debug log : %s₩n", szLog );
}

int main()
{
        // Log callback 함수등록
        SetCallbackFunction( CALLBACK_LOG, (PVOID)CS3LogCallback );
}
```

나. CS인증 3.0 의 업데이트 정보를 전달 받기 위해서 callback 함수를 등록합니다.

```
void __stdcall CS3UpdateInfoCallback( int nUpdateType, int nBefore, int nAfter )
{
        if( nUpdateType == UPDATE_PROTOCOL )
          printf("this is protocol update notification : %d -> %d ₩n", nBefore, nAfter);

        else if( nUpdateType == UPDATE_GGVERSION )
```

```
        printf("this is ggversion update notification : %d -> %d ₩n", nBefore, nAfter);
}

int main()
{
        // Update callback 함수등록
        SetCallbackFunction( CALLBACK_UPDATE, (PVOID)CS3UpdateInfoCallback );
}
```

다. Hack report 가 클라이언트에서 전달 된 경우 DecryptHackData 함수로 복호화가 가능합니다.
(Hack report : 클라이언트의 NPGameMonCallback 함수에서 NPGAMEMON_GAMEHACK_REPORT 로 전달되며, 게임사 자체적으로 핵툴 통계를 낼 경우에 사용 됩니다. 모든 핵툴이 report 정보를 전달하지는 않습니다.)

```
int main()
{
        // Hack report 가 클라이언트에서 전달된 경우
        UINT32 uReceivedSize = RecvFromClient( pbHackReport );

        // GameName 은 게임클라이언트에서 CNPGameLib 의생성자에 전달된 게임이름을 말합니다.
        UINT32 uReturn = DecryptHackData( "GameName", pbHackReport, uReceivedSize );
        if( uReturn == -1 )
        {
                // 실패
                return;
        }
}
```

라. 선택적으로 어떤 유저에서 에러가 발생하였는지 알고 싶을 경우 SetUserInfo()함수를 호출하면 에러 발생시 서버로그에 유저의 ID, IP Address 등의 정보를 기록합니다. (7. 함수설명 참고)

- 로그 기록 예:
[ObjectFunction] ErrorCode insert : 3302, UserId: GameId, User IPAddr: 127.0.0.1, User ExtendInfo: ETC

```
class CUser
{
        ...
public:
        CCSAuth3 csa;
}

int main(void)
{
        CreateUser(&pUser);
        pUser->csa.Init( true );
        // User별 Init 후에 호출합니다.
        UINT32 uReturn = pUser-> csa.SetUserInfo( INFO_ID, "GameId" );
        uReturn = pUser-> csa.SetUserInfo( INFO_IPADDR, "127.0.0.1" );
        uReturn = pUser-> csa.SetUserInfo( INFO_EXINFO, "ETC" );
}
```

④ 클라이언트 적용 (필수 구현)

가. 서버에서 CS인증 3.0 을 전송 받았을 때 게임가드에게 바로 전달합니다.
(Auth3 의 세번째 인자인 DWORD dwServerNumber 은 서버 이동시에 안정성 확보를 위해 사용됩니다. 일반적으로 사용하지 않고 아무 숫자나 입력해도 됩니다.)
(참고4 : 서버 이동시 주의점과 dwServerNumber 사용방법)

```
// 서버로부터 인증 패킷을 수신
UINT32 uReceivedSize = RecvFromServer( packet );

// 게임가드로전달
// dwServerNumber 은 일반적으로 사용하지 않아도 무관합니다.
npgl.Auth3( packet, uReceivedSize, 0 );
```

나. 게임가드의 Callback 함수인 NPGameMonCallback 에서 CS인증 3.0 응답 값을 받으면 그대로 서버에 전송합니다.

```
// 게임가드 CALLBACK 함수
BOOL CALLBACK NPGameMonCallback(DWORD dwMsg, DWORD dwArg)
{
    switch (dwMsg)
    {
        case NPGAMEMON_CHECK_CSAUTH3:
        {
            PCSAuth3Data pCSAuth3 = (PCSAuth3Data)dwArg;
            SendToServer( pCSAuth3->bPacket, pCSAuth3->dwPacketSize );
            break;
        }
    }
}
```

⑤ C 전용 적용

- ggsrv30.h 에서 #define _HEADER_C 를 주석해제 합니다.
- C 에서는 CCSAuth3 class 를 사용할 수 없으므로 C 전용 함수를 사용합니다.
- 나머지 함수는 동일 합니다. (InitCSAuth3, CloseCSAuth3 등..)

```
        // 모든C 지원함수는 내부에서 CCSAuth3 객체의 메모리 유효성 여부를 확인합니다.

        // CCSAuth3  객체생성 (내부에서 malloc 을 호출)
        PBYTE pCSInstance = CSAuth3_CreateInstance();

        // CCAuth3::Init() 과 동일
        uReturn = CSAuth3_Init( pCSInstance, TRUE );

        // CCSAuth3::Get() 과 동일
        uReturn = CSAuth3_Get( pCSInstance, &uPacketSize );

        // CCSAuth3::packet 을return 함 (csa.packet 대신 사용)
        PBYTE packet = CSAuth3_GetPacket( pCSInstance );

        // CCSAuth3::Check() 과 동일
        uReturn = CSAuth3_Check( pCSInstance, packet, uPacketSize );
```

```
// CCSAuth3::Close() 와 동일
uReturn = CSAuth3_Close( pbUser );

// CCSAuth3 객체를소거함(내부에서free 를호출)
CSAuth3_Release( pbUser );
```

⑥ Linux 환경에서의 적용

- 대부분 C-Runtime Library 를 이용하였기 때문에 윈도우 환경에서의 적용과 별다른 차이점이 없습니다. 다만 gcc 의 버전을 3.x 와 4.x 를 구분하여 라이브러리를 제공해 드리므로 버전에 맞추어서 적용을 하십시오.

- Dynamically Loaded (DL) 라이브러리를 사용할 경우에는 컴파일 시에 -ldl 옵션을 사용하셔야 합니다. (되도록이면 Dynamic loaing 방식(.so 파일 사용) 을 사용할 것을 부탁드립니다. 문제점 발생시 빠른 대응이 가능하도록 대부분의 기능을 .so 파일로 옮기고 버전 관리 방식도 변경하였으나, static loading 방식으로 게임 서버가 운영이 된다면 서버 재컴파일 없이는 업데이트를 할 수 없기 때문에 지원에 어려움이 있습니다.)

- C 컴파일러를 사용하는 경우, C++ 관련 링크 에러가 나면 -lstdc++ 옵션을 사용하셔야 합니다.

- 동기화를 처리하기 위해 -pthread 옵션을 사용하셔야 합니다.

⑦ Win64 환경에서의 적용

Win64 전용 모듈을 사용하고, 구현 방법은 Win32 와 동일합니다.

⑧ Delphi 등 다른 언어에서 구현

- ggsrv30.h 에서 #define _HEADER_DLL 을 주석 해제 합니다.
- ggauth30lib.lib 대신 ggauth30lib.dll 을 사용하고 dll 을 직접 Load 하여 사용하시기 바랍니다.

4. 추가 기능

- 서버인증 3.0 을 통해서 제공되는 부가적인 기능에 대한 설명입니다.

① 구버전 클라이언트 차단

- 오래된 게임 클라이언트를 사용하는 것을 막도록 서버인증을 통해서 제공되는 기능입니다. 수동 차단방식과 자동 차단방식 두가지 방식을 제공합니다. csauth3.cfg 설정파일을 통해서 기능을 선택할 수 있으며 세부적인 설정 방법은 csauth3.cfg 의 7. CLIENT 항목에 설명 되어 있습니다.

▪ 수동 차단 방식

- 게임 클라이언트의 실행파일(exe) 을 서버에 복사하여 복사되어 있는 파일만 실행을 허용 하는 방식입니다.

- 장점

* 직관적인 방법으로 가장 정확하게 구버전을 차단할 수 있습니다.

- 단점

* 서버인증 3.0 을 적용한 모든 서버에 최신 게임 클라이언트를 항시 복사해주어야 하는 번거로움
* 관리 실수가 발생할 경우 대량의 에러가 발생할 가능성이 존재함

- 사용 방법

1. CheckClientType=1 설정
2. ClientDirectory 에 클라이언트를 복사할 폴더 이름 저장 (기본 Client)

3. ggauthxx_xx.dll 과 동일한 폴더에 2번에서 지정한 폴더 만듦 (대소문자 구분)
4. 해당 폴더에 허용할 게임 클라이언트 복사
5. 서버 실행

- **게임 클라이언트 업데이트 방법**

* 서버를 재시작 하지 않더라도 서버 실행 중 언제라도 지정된 폴더에 클라이언트를 복사만 해준다면 자동으로 허용 할 클라이언트가 업데이트 됩니다.

1. 위에서 지정한 폴더에 새로운 클라이언트 복사
2. 구버전 클라이언트를 삭제
3. 클라이언트 정보의 업데이트 주기는 ClientReloadMinutePeriod 에서 설정 (분 단위)
4. 새로운 클라이언트가 등록되었을 시에 UpdateLog 에 업데이트 정보가 남음

- **주의 사항**

1. 클라이언트 실시간 등록 시 등록 시간 지연 가능성
- 새로운 클라이언트를 실시간으로 등록하는 것은 경우에 따라서 ClientReloadMinutePeriod 에서 설정된 시간의 2배가 걸리는 경우가 있을 수 있습니다.
(서버인증에서 클라이언트 업데이트 하는 도중에 동시에 게임사에서 파일을 추가하고 삭제하여 내부적으로 파일 open을 실패하는 경우)
그러므로 서버의 파일등록은 실제로 엔드유저에게 업데이트 하기 전에 충분한 시간을 가지고 교체를 해 주시고 정상등록 여부를 미리 게임 실행을 통해서 확인 해주는 것이 안전합니다. 업데이트의 성공 여부는 서버인증 UpdateLog 를 통해서도 실시간으로 확인 할 수 있습니다.

2. 악성 프로그램으로 인한 오진의 가능성
- 클라이언트의 구분은 파일을 기반으로 합니다. 파일이 바이러스나 악성 프로그램에 의해서 변조되는 경우 악의적인 의도가 없음에도 허용되지 않는 클라이언트로 구분되어 서버인증에서 차단할 수 있습니다. 이 경우를 대비하기 위해서 실행파일의 코드 영역만을 구분 값으로 설정하는 옵션(CodeSectionOnly=1) 을 제공하고 있습니다.

▪ 자동 차단 방식

- 게임 클라이언트를 구분할 수 있는 정보를 서버에서 취합하고 통계내어 다수가 사용하는 최신 클라이언트를 제외한 나머지 구버전 클라이언트를 사용하는 유저를 차단하는 방식입니다.

- **동작 원리**

* 서버에서는 현재 사용중인 각각 게임 클라이언트의 사용자 수를 관리하는 테이블이 존재합니다.
유저는 게임을 실행하면 실행된 클라이언트를 구분할 수 있는 고유 정보 (게임 클라이언트 Hash) 를 서버로 전달하며, 서버는 해당 정보의 사용자 수를 증가시키며 접속을 끊을 시 해당 사용자 수를 감소하는 식으로 동접자 수를 관리합니다.

매 5분 (UpdateMinutePeriod=5) 마다 한번식 테이블을 조사하여 80% (WhitePercent=80) 이상의 유저가 사용하는 클라이언트가 존재한다면 정상적인 최신 클라이언트로 판단하여 WhiteList 에 등록합니다. 그리고 3% (BlackPercent=3) 이하의 유저가 사용하는 클라이언트가 존재한다면 구버전 혹은 비정상적은 클라이언트로 판단하여 BlackList 에 등록하고 이후 해당 클라이언트로 접속할 경우 차단을 합니다.

WhiteList 에 등록된 클라이언트의 수가 (AllowClientCount=1) 보다 많을 경우 가장 오래전에 등록된 클라이언트를 구버전 클라이언트로 판단하여 BlackList 로 등록하고, 이 후 해당 클라이언트 접속에 대해서 차단을 합니다.

처음 실행되는 클라이언트의 정보가 서버로 전달된다면 서버는 2시간 (ActivationMinutePeriod=120) 동안 판단을 유보하고 대기한 이후에 Black 의 조건여부를 판단합니다.

너무 적은 유저가 사용중인경우 White/Black 의 비율정보가 왜곡 될 수 있기 때문에 이를 판단하기 위한 최소 동접자 수를 100명 (ActivationUserNum=100) 으로 제한 하였습니다.

업데이트 된 BlackList 는 새로 접속하는 유저만 차단을 하며 이미 게임중인 유저는 차단하지 않습니다.

- **장점**

* 최신 게임 클라이언트를 서버로 복사하는 번거로움이 없이 자동으로 차단이 가능함

- **단점**

* 일부 구버전이 사용 될 수 있는 홀이 존재함.
* 각 게임사의 환경에 따라서 설정을 변경해주어야 할 가능성이 있음

- **사용 방법**

1. csauth3.cfg 의 CLIENT 항목 설정 방법 참조
2. csauth3.cfg 의 CheckClientType=2 를 설정
3. 특수한 경우 이외에는 기본 설정 사용
(csauth3.cfg 의 설정값을 변경하면 30초 (ConfigAutoLoadPeriod=30) 내로 설정이 반영됨)

- **정상적인 최신 클라이언트가 차단되는 경우 긴급조치 방법**

1. TurnOffCheckClient=1 로 해당 기능을 끔
2. UpdateLog 에서 테이블 정보 확인하고 White/Black 의 수치를 적절하게 변경
3. TurnOffCheckClient=0 으로 변경하고 서버 재시작

- **주의 사항**
- 다음 상황은 일반적이지 않은 특수한 경우이므로 문제가 될 경우 적절한 옵션 수정이 필요합니다. UpdateLog 에 테이블 정보가 주기적(LogHourPeriod=6 6시간마다)으로 남으므로 이를 참조하시기 바랍니다.

1. 구버전 클라이언트 사용자가 너무 많아서 정상 클라이언트 사용자가 80% 가 안될 경우 정상적인 차단을 못할 수 있습니다.

2. 특정 구버전 클라이언트가 사용자가 너무 많아서 동접자 3%를 넘을 경우 해당 특정 구버전 사용자가 차단이 되지 않을 수 있습니다.

3. 새로운 클라이언트가 등록된 이후에 2시간 동안 동접 3%를 확보하지 못한 경우 정상 클라이언트가 BlackList 로 등록되어 차단 될 수 있습니다.
(신규 접속이 거의 없는 세벽에 업데이트 된 경우 발생 가능성이 있음)

5. csauth3.cfg 파일 설정법

- 서버인증 3.0 의 모든 설정은 csauth3.cfg 에서 설정할 수 있으며, 따로 변경할 수 있는 함수는 제공되지 않습니다. 아래의 설명을 참조하여 각 게임사의 상황에 맞게 설정을 변경하여 사용하시면 됩니다. 기본설정 값을 사용하시면 무난 합니다.
- csauth3.cfg 는 Windows 의 ini 파일 문법을 사용합니다.

① CONFIG
- CS인증 3.0 의 기본적인 설정

*** ConfigAutoLoadPeriod=30**
- 지정된 시간(초) 주기로 csauth3.cfg 설정 파일을 읽어들여 정보를 반영합니다. 설정을 변경하더라도 최대 위의 설정시간이 지난 이후에 반영 됩니다. 파일을 다른 프로그램이 Open 하여 CS인증 에서 Open 하는데 실패하더라도 에러가 발생하지 않도록 처리 되어 있습니다.

② UPDATE
- 프로토콜, 내부버전의 업데이트 조건 설정 (CS인증 2.x의 SetUpdateCondition 과 동일)

* MinLastestModuleUserCount=30
* StatisticsResetPeriod=15
* PercentageForUpdate=50
- (참고6 : 버전 업그레이드 조건의 동작 방식)

③ TURNOFF
- CS인증 ON/OFF

*TurnOffCSAuth=0
- 0 인 경우 CS 인증 사용(NOT TurnOff)을 의미하고
1 인 경우 CS 인증을 끄는 것(TurnOff)을 의미합니다.
CS인증을 끄는 경우 CS인증의 모든 함수가 ERROR_SUCCESS(0) 을 return 하게 됩니다.
긴급하게 CS 인증을 꺼야만 하는 경우가 있을 때 사용합니다.
(주의 : 서버인증을 끄기 위해서 0 에서 1로 변경하는 것만 가능하며 1 에서 0 으로 변경하는 것은 불가능 합니다. 때문에 서버인증을 다시 켜기 위해서는 옵션을 변경하고 서버를 재시작 해야만 합니다.)

- 세부 기능 OFF
*TurnOffSuspend=0
*TurnOffAlgorithm=0
*TurnOffUnique=0
*TurnOffCheckClient=0
- 서버인증 내의 세부 기능을 막는 옵션 으로 장애가 발생하였을 시에 사용됩니다.
임의로 옵션을 활성화 시키는 것은 권장하지 않습니다.

④ PATH
- CS인증이 사용하는 각종 폴더 설정
- 서버의 CurrentDirectory 를 기준으로 폴더 경로를 설정합니다. 해당 폴더가 존재하지 않는 경우 상위폴더의 유무와 상관없이 폴더가 자동생성 됩니다.

* UseCurrentDirectory=0
- 모든 경로는 InitCSAuth3 을 입력받은 경로를 기준으로 생성되나 이 옵션을 사용할 경우 서버의 CurrentDirectory 기준으로 생성된다.

* DataBase=DataBase
- CS인증에서 사용하는 각종 정보를 저장하는 폴더, 몇kb 수준의 소량의 용량이 사용됩니다.

* NormalLog=Log/NormalLog
* UpdateLog=Log/UpdateLog
* DebugLog=Log/Debug
- 각각의 Log 가 생성됩니다. 서버가 실행되는 연/월/일/시/분/초 이름을 가지는 파일이 생성되며 오래된 파일을 삭제하지 않으므로 서버관리자의 적절한 관리가 필요합니다. CS 인증 문제 발생시에 이 Log 파일을 저희 게임가드 측으로 보내주시면 문제 발생시에 원인 분석을 해드릴 수 있습니다.

* ClientDirectory=Client
- 최신 클라이언트를 등록할 폴더 경로

⑤ LOG
- Log 의 각종 설정 (0 - 사용안함, 1 - 사용)

* UseNormalLog=1
* UseUpdateLog=1
* UseDebugLog=0
- 각각의 로그의 사용 여부를 설정합니다. DebugLog 는 서버에 부하를 줄 수 있으며, 용량도 크기 때문에 일반적으로 0 을 설정을 하고, 게임가드 측의 요청이 있을 시에 1 로그를 추가로 남겨 주시기 바랍니다.

* CloseAfterNormalLogWrote=1
* CloseAfterUpdateLogWrote=1
* CloseAfterDebugLogWrote=0
- 로그 한줄 기록후 파일을 Close 할 것인지 결정
- 한줄 기록시마다 파일을 Close 할 경우 share 여부와 상관없이 서버 동작중에 파일사용이 용이해지지만 로그 기록의 빈도가 높다면 부하가 일어날 수 있습니다. UpdateLog 와 NormalLog 는 기록의 빈도가 높지 않으므로 1 을 주어도 무리가 없을 것 입니다.

⑥ CLIENT
- 구버전 클라이언트 차단 기능
(4-1. 구버전 클라이언트 차단기능 참조)

* CheckClientType=0
- 0 : 구버전 클라이언트 차단 기능을 사용하지 않음
- 1 : 구버전 클라이언트 수동 차단
- 2 : 구버전 클라이언트 자동 차단

* ClientReloadMinutePeriod=10
- ClientDirectory 에 존재하는 최신 클라이언트 업데이트 주기 (분)

* CodeSectionOnly=1
- 실행파일이 바이러스에 변조되는 경우 발생하는 오진을 막기 위한 옵션으로 코드 영역만을 구분값으로 사용함

* AllowClientCount=1
- 허용할 최신 클라이언트 개수

* WhitePercent=80
* BlackPercent=3
- WhiteList/BlackList 에 등록될 클라이언트 동접자 비율

* ActivationUserNum=100
- 해당 기능 활성화를 위한 최소 접속자 수

* ActivationMinutePeriod=120
- 처음 등록된 클라이언트 정보의 활성화 시간 (분)

* UpdateMinutePeriod=5
- 서버의 클라이언트 정보관리 테이블 모니터링 주기 (분)

* LogHourPeriod=6
- 게임 클라이언트 상태 테이블 정보를 UpdateLog 에 남기는 주기 (시)

⑦ DEBUG

6. 참고

① **권장 주기**
- CS 인증 3.0 의 권장 주기는 3~5분 입니다. 1분 이하가 될 경우 CS 인증 3.0 의 동작 방식에 의해서 게임가드가 정상동작하지 않고 멈추어 있는 것으로 판단하고 에러를 발생시킬 소지가 있습니다. 또한 주기가 10분 이상이 된다면 마찬가지로 멈추어 있는 것으로 판단할 가능 성이 있습니다. 최소 1분 30초, 권장 주기는 3~5분 입니다.

② **timeout 구현 금지**
- 자체적으로 timeout 을 구현해서 클라이언트에서 응답이 오지 않는 경우 접속을 끊도록 서버를

구현하는 경우가 많습니다. 이렇게 구현하는 것을 자제해 주실 것을 부탁드립니다. CS 인증은 자체적으로 Get() 함수가 Check() 함수 호출이 없이 2회 연속 호출 될 경우 응답 없음 에러를 발생시키기 때문에 주기마다 Get() 함수만 호출해 주시면 됩니다.

- timeout 을 자체적으로 구현할 경우 다음과 같은 문제점이 발생합니다

- CS 인증 3.0 은 2.6 과 마찬가지로 게임가드에서 일정한 시간 지연을 가지고 응답합니다. 그리고 특정 상황에는 응답이 조금 더 지연되는 문제가 발생할 수도 있습니다. 이런 경우 정상 유저가 게임사에서 구현한 timeout 에 의해서 접속 끊김이 발생할 수 있습니다.

- 게임가드 사내에서 핵툴 분석 및 디버깅을 위해서 한 PC 에서 임의로 CS 인증을 끄는 경우가 있습니다. 이런 경우 timeout 이 구현이 되어 있다면 그로 인한 접속 끊김 때문에 핵툴 분석에 어려움이 발생할 수 있습니다.

- timeout 에 의한 접속 끊김은 CS 인증 자체의 에러통계에 포함되지 않기 때문에 문제 발생시에 대응이 어려울 수 있으며, 에러로그서버의 통계에도 포함되지 않습니다. 지원시에도 로그에 정보가 표시 되지 않으므로 지원이 힘든 측면이 있습니다.

③ 로그의 종류

- 로그는 3가지 종류가 존재합니다.

- Update Log
  - 프로토콜, 내부버전이 업데이트 조건을 충족하여 업데이트 된 경우에 로그를 남깁니다.

- Normal Log
  - 초기화 상황, 일반적인 상황, 에러발생시 코드 등의 로그와 Update Log 의 로그를 남깁니다.

- Debug Log
  - 문제 발생 시에 GameGuard 측에서 디버깅을 하기 위해서 최대한 자세한 로그를 남깁니다. 이 중에 모든 유저의 패킷 로그도 남기게 되므로 일반적인 상황에서는 절대로 남기도록 설정해서는 안됩니다. (csauth3.cfg 에서 UseDebugLog=0 으로 설정하면 callback 도 발생하지 않음) Normal 로그 와 Update 로그도 포함 됩니다.

④ 서버이동시 주의점과 dwServerNumber 사용 방법

- 유저가 현재 서버에서 CS 인증을 하다가 다른 서버로 이동해서 다시 CS 인증을 하는 경우 여러가지 변수에 의해서 문제가 발생 할 수 있습니다.

- 신규 게임가드 모듈 업데이트 시에 서버 이동 시에 서버간의 내부버전의 차이가 발생하여 정상 접속 유저가 구버전 유저로 판단되어서 접속 끊김이 발생할 수 있습니다. CCSAuth3::Init(bool bIsFirstServer) 함수에 초기화 시에 서버 이동을 한 유저에 대해서는 bIsFirstServer 를 false 로 준다면 구버전 유저에 대해서 접속을 끊지 않습니다. 다만 이전 서버에서 1회 응답을 주고 받음으로 구버전 확인을 한 유저에 한해서만 적용되어야 하므로 구현이 복잡한 측면이 있습니다. 이 이슈가 문제가 되는 경우에만 조건확인 이후에 bIsFirstServer 변수를 이용한다면 해결이 가능 할 것으로 보입니다.

- 서버 이동을 하고 즉시 CCSAuth3::Init() 을 호출하고 첫번째 인증을 시작한다면 게임가드 콜백의 응답시간이 20초 까지도 지연되는 특성 때문에 이전 서버의 응답값을 다음 서버로 전송하게 되어 응답 에러가 발생할 수 있습니다. **서버이동을 한 후에 1분 정도 지나서 CS 인증을 시작한다면 이 문제를 피해갈 수 있습니다.**

- 서버 이동 이후에 빠른 차단을 위해서 1분도 기다릴 수 없는 경우 Auth3함수의 dwServerNumber 을 이용해서 문제를 해결 할 수 있습니다. dwServerNumber 은 Auth3() 에 입력한 값이 NPGameMonCallback 에 그대로 전달됩니다. 이런 특성으로 시간적 Gap 이 있는 Auth3 과 NPGameMonCallback 의 짝을 맞출 수 있습니다. dwServerNumber 을 이용한 서버이동 문제 해결 방법의 구현은 다음과 같습니다.

```
DWORD g_dwCurrentServerNumber = 0;
```

```
// CS 인증값이 오는 경우 Auth3 호출 중..
npgl.Auth3( ... , g_dwCurrentServerNumber );
...
npgl.Auth3( ... , g_dwCurrentServerNumber );
...

// 서버이동 발생시 ServerNumber 를 1 더함
g_dwCurrentServerNumber++;

npgl.Auth3( ... , g_dwCurrentServerNumber );
...
npgl.Auth3( ... , g_dwCurrentServerNumber );
...

BOOL CALLBACK NPGameMonCallback(DWORD dwMsg, DWORD dwArg)
{
        case NPGAMEMON_CHECK_CSAUTH3:
        {
           PCSAuth3Data pCSAuth3 = (PCSAuth3Data)dwArg;

           // 이 CS 인증이 지금 서버에서 보낸 응답일 경우에만 서버에 전송 (아니면 폐기)
           if( pCSAuth3->dwServerNumber == g_dwCurrentServerNumber )
              SendToServer( pCSAuth3->bPacket, pCSAuth3->dwPacketSize );

           break;
        }
}
```

- 위의 구현 방법 대신에 클라이언트에서 현재 접속한 서버를 구분짓는 특별한 값이 따로 있다면, 그 값을 사용하시는 것도 괜찮습니다.

⑤ 에러코드 종류와 의미

- 에러코드 3000 이상 접속 해제의 의미
  - 에러코드 2900 번대는 WARNING 을 의미합니다. 초기화를 하지 않고 Get 함수를 호출하는 경우나 프로토콜 업데이트 시에 메모리 부족으로 인한 로딩 실패등 사용자의 잘못이 아니라 시스템의 오류나 개발자의 실수에 의해서 발생하게 됩니다. 이 경우에도 유저의 접속을 끊는 것은 큰 문제를 유발 시킬 소지가 있기 때문에 접속해제 없이 추후에 로그를 확인하여 2900번대 오류가 다수 발생한 경우 해당 문제점만 따로 해결 하는 것이 좋습니다.

- 에러코드 번호별의 의미
  - 2900 번 대 : WARNING - 시스템의 오류나 개발자의 실수로 인한 에러 상황
  - 3000 번 대 : CS 인증 개발시에 발생 할 수 있는 구현 미스에 의한 오류 상황
  - 3100 번 대 : 시스템이 발생시키는 오류로서 메모리 에러가 대표적
  - 3200 번 대 : CS 인증을 올바르지 않게 사용하여 발생하는 게임서버 개발자 미스에 의한 에러
  - 3300 번 대 : 서버에서 발생하는 일반적인 유저 접속해제가 필요한 상황
  - 3400 번 대 : 서버에서 발생하는 내부 알고리즘에 의한 유저 접속해제가 필요한 상황
  - 3500 번 대 : 3400대와 동일
  - 3600 번 대 : 클라이언트에서 발생하는 유저 접속해제가 필요한 상황

  * 자세한 에러코드에 대한 설명과 대응 방안은 **8. 에러코드의 자세한 설명과 대응방법** 을 참고하시기 바랍니다.

⑥ 버전 업그레이드 조건의 동작 방식

- CS 인증 2.5/2.6 의 SetUpdateCondition 함수와 동일한 의미를 가지는 변수 입니다. 대신 알고리즘 이 수정되어 동작 방식은 변경 되었습니다.

* MinLastestModuleUserCount=30
* StatisticsResetPeriod=15
* PercentageForUpdate=50

- 프로토콜과 내부버전 모두 동일한 알고리즘을 통해서 업데이트를 진행하게 됩니다.

- 서버의 현재 버전과 같거나 높은 접속자의 첫번째 인증값이 응답 시에 통계에 추가 됩니다.

- StatisticsResetPeriod (분) 시간 마다 통계 값을 이용하여 업데이트 여부를 확인하고 통계값을 초기화 합니다.

- 업데이트 확인 시간에 총 통계의 수 중에서 PercentageForUpdate (%) 을 넘는 버전이 있는 경우 업데이트 대상이 됩니다.

- PercentageForUpdate 를 만족하고 최신 버전이 MinLastestModuleUserCount 를 넘는 다면 **서버 내부의 버전을 업데이트 하고 이후에 보다 낮은 버전의 접속자를 차단**합니다.

- 위의 예시는 15분 주기의 통계동안 통계 대상의 총합중에 50% 를 넘는 최신 버전이 있으면서 사용자 수가 30명 이상인 경우 그 버전으로 업데이트하고 없다면 통계치 초기화 하고 다시 통계를 시작한다는 의미입니다.

- 이처럼 복잡한 조건을 만족해야 업데이트를 진행하는 이유는 첫째 잘못된 버전 사용에 의한 리스크를 최소화 하고, 신규 접속자가 최신 버전을 받는 시간적 여유를 줌으로서 서버의 부하를 분산시키기 위함입니다.

⑦ CS 인증 2.5 / 2.6 에서 변경 시 주의사항

- 첫 인증부터 3~5분 주기를 적용합니다. 2.5 / 2.6 에서는 첫인증과 두번째 인증을 붙여서 호출 하도록 가이드 되었으나 3.0 에서는 그렇게 적용한다면 에러코드가 발생하게 됩니다.

7. 함수 설명

- **UINT32 InitCSAuth3(char *szModuleDir)**
  - CS인증 3.0 을 초기화 합니다.
  - 초기화 하지 않고 다른 함수 또는 맴버함수를 호출할 경우 에러코드가 발생합니다.

szModuleDir	모듈 파일(.dll, .so) 과 csauth3.cfg 가 있는 폴더를 입력, 끝의 slash 유무는 상관 없음
return	ERROR_SUCCESS(0) - 성공 not 0 - 에러코드

- **UINT32 CloseCSAuth3()**
  - CS인증 3.0 을 Close 합니다.
  - DataBase 폴더에 CS인증 3.0 의 정보를 기록하므로 서버 종료시에 반드시 호출 되어야 합니다.

return	ERROR_SUCCESS(0) - 성공 not 0 - 에러코드

- **UINT32 DecryptHackData(char *lpszUserKey, LPVOID lpData, DWORD dwLength)**
  - 게임 클라이언트에서 Hack report 로 전달된 암호화된 문자열을 복호화 합니다.
  - 게임가드의 Hack report 정보를 자체적으로 수집하는 경우에만 사용 됩니다.

lpszUserKey	Key 값으로 사용하는 게임클라이언트에서 CNPGameLib 의 생성자에 입력된 게임이름을 입력함
lpData	전달 받은 암호화된 문자열을 입력
dwLength	전달 받은 암호화된 문자열의 길이를 입력
return	(UINT32)-1 : 실패 not (UINT32)-1 : 복호화된 문자열의 길이

- **UINT32 SetCallbackFunction(int nFunctionType, PVOID pFunction)**
  - 로그 callback 함수와 업데이트 callback 함수를 등록합니다.
  - 필요에 의현 경우에만 등록하고 등록하지 않더라도 CS인증 동작에 영향을 미치지 않습니다.

nFunctionType	등록할 callback 함수의 유형을 입력 (enum CallbackType 중 1)
pFunction	위의 유형의 함수를 구현하고 함수 포인터를 입력 (PCS3LogCallback, PCS3UpdateInfoCallback 형의 함수포인터 입력)
return	ERROR_SUCCESS(0) - 성공 not 0 - 에러코드

- CCSAuth3
  - 유저별 CS인증 3.0 클래스 로서 서버에 접속한 모든 유저는 CCSAuth3 객체를 1개씩 생성하여 보유해야 합니다.

- UINT32 CCSAuth3::Init(IN BOOL bIsFirstServer)
  - 유저별 CS인증 3.0 클래스 초기화 함수입니다.
  - 인증 Sequence 를 비롯한 모든 data 가 초기화 됩니다. 클래스 첫사용이나 재사용시에 반드시 호출 되어야 합니다. 호출 되지 않는다면 다른 맴버함수에서 에러코드가 발생합니다.

bIsFirstServer	일반적인 상황에서 항상 true, 5. 참고 의 4번 항목 참고
return	ERROR_SUCCESS(0) - 성공 not 0 - 에러코드

- UINT32 CCSAuth3::SetUserInfo(IN UINT32 uUserType, IN char* szUserInfo)
  - 유저별 정보를 넘겨주는 함수 입니다.
  - 해당 함수를 호출하면 에러발생시 유저의 정보를 서버로그에 기록합니다.
  - UINT32 CCSAuth3::Init(IN BOOL bIsFirstServer) 함수 호출 후에 호출합니다.
  - uUserType마다 szUserInfo 버퍼의 길이 제한이 있습니다.
    INFO_ID는 32byte, INFO_IPADDR는 24byte, INFO_EXINFO는 64byte입니다.

uUserType	유저정보의 타입입니다. (INFO_ID=0,INFO_IPADDR=1,INFO_EXINFO=2)
szUserInfo	각 타입별 맞는 정보를 담습니다.
return	ERROR_SUCCESS(0) - 성공 not 0 - 에러코드

- UINT32 CCSAuth3::Close()
  - 유저별 CS인증 3.0 클래스 종료 함수로서 사용자의 접속 해제시에 동작하여야 할 기능이 다수 포함되어 있습니다.
  - 반드시 호출이 되도록 CCSAuth3 클래스의 소멸자에서도 Close 를 호출하고 있고 Init

이후에 Close 호출 없이 다시 init 이 호출 되는 경우 내부에서 Close 를 호출 하고 있습니다.

return	ERROR_SUCCESS(0) - 성공 not 0 - 에러코드

- UINT32 CCSAuth3::Get(OUT UINT32 *puSize)
  - 맴버변수인 packet[4096] 에 패킷을 생성하고 패킷 크기를 puSize 에 담습니다.
  - Get 함수에서 지정시간 마다 csauth3.cfg 파일을 load 하므로 Get 함수가 호출 되지 않는다면 csauth3.cfg 설정도 적용되지 않을 수 있습니다.

puSize	생성된 패킷의 크기를 담습니다. (OUT)
return	ERROR_SUCCESS(0) - 성공 not 0 - 에러코드

- UINT32 CCSAuth3::Check(IN UINT32 uSize)
  - 응답 패킷의 유효성을 검사합니다.
  - packet[4096] 에 응답 패킷을 복사하고 uSize 에 패킷 크기를 담아주어야 합니다.

uSize	입력한 패킷의 크기를 전달 합니다. (IN)
return	ERROR_SUCCESS(0) - 성공 not 0 - 에러코드

- CCSAuth3::uMagic
  - C 버전 함수 사용시에 CSAuth3_CreateInstance() 에서 전달 받은 객체 포인터를 올바르게 입력하고 있는지 확인하기 위한 메모리 시그너쳐

- CCSAuth3::packet
  - Get / Check 함수는 packet 맴버 변수를 통해서만 데이터를 입출력 받습니다. 메모리 복사가 1회 발생하여 효율이 떨어질 수는 있지만. 혹시 발생할 수도 있는 메모리 에러에 대한 분쟁의 소지를 없애기 위해서 따로 변수를 두었습니다.

- CCSAuth3::data
  - CS 인증을 위한 유저별 맴버변수

8. 테스트

- 테스트는 아래의 방법으로 진행 할 수 있으며 아래의 확인 사항을 충족할 경우 CS 인증이 정상 적용된 것으로 볼 수 있습니다.

1. 테스트를 진행할 테스트 서버를 따로 구축 (존재하는 경우 기존 테스트 서버 이용)
2. 게임가드 측은 CS 인증 3.0 에 대응하는 게임가드 모듈 제공하여 적용
3. 서버 실행시 InitCSAuth3 함수가 ERROR_SUCCESS(0) 를 return 하는지 확인
4. 유저 접속시 CCSAuth3::Get() 함수가 ERROR_SUCCESS(0) 를 return 하는지 확인
5. 수신한 패킷을 CCSAuth3::Check() 에 입력시 ERROR_SUCCESS(0) 를 return 하는지 확인
6. csauth3.cfg 파일에서 TurnOffCSAuth=1 을 설정하여도 10분 이상 정상적인 게임실행이

가능한지 확인

7. 에러코드 발생시 8. 에러코드 설명과 대응 방법 을 참조하여 문제 해결

8. 정보가 부족할 경우 게임가드 측에 에러코드와 클라이언트의 erl, 서버의 Normal 혹은 Debug 로그를 제공하여 문의
(테스트 서버 기동 시 클라이언트를 게임가드측에 제공해 주시면 추가적으로 WARNING 임의 발생, 디버그 상황 임의 발생 등을 추가 테스트 할 수 있습니다.)

9. 에러코드 설명과 대응 방법

- 2900 번대 에러 : 경고 - 시스템의 오류나 개발자의 실수로 인한 에러 상황으로서 유저의 접속 해제를 할 상황은 아닙니다. Normal Log 확인 시 2900 번대 에러가 다수 발생한다면 서버 코드의 수정이 필요합니다.

2900	▪ WARNING_GET_BEFORE_INIT ▪ CCSAuth3::Init 을 호출하기 전에 Get을 호출하는 경우 ▪ 먼저 Init 이 호출 되도록 코드를 수정
2901	▪ WARNING_CSAUTH3_TURNOFF ▪ 게임가드 측에서 핵툴분석, 디버깅등의 이유로 서버인증을 개별적으로 off 한 경우
2902	▪ WARNING_CHECK_BEFORE_INIT ▪ CCSAuth3::Init 을 호출하기 전에 Check 을 호출 하는 경우 ▪ 먼저 Init 이 호출 되도록 코드를 수정
2903	▪ WARNING_CHECK_WITHOUT_GET ▪ Get 함수가 호출 되지 않고 Check 이 먼저 호출 됨 ▪ 서버이동 한 후 새로운 패킷 전송을 하기 전에 이전 서버에서 받았어야 할 응답을 받은 경우
2904	▪ WARNING_OBJECT_CALL_BEFORE_INIT ▪ InitCSAuth3 을 호출 하지 않고 CCSAuth3 의 맴버함수를 호출 한 경우 ▪ 다른 모든 함수보다 InitCSAuth3 이 먼저 호출 되도록 수정
2905	▪ WARNING_INVALID_PROTOCOL_REQUEST ▪ ggauth32_x 모듈이 보유하지 않은 프로토콜이 요청됨 ▪ 있을 수 없는 상황이나 해커, 혹은 CS인증 버그에 의해서 발생할 수 도 있음 게임가드 측에 문의
2906	▪ WARNING_NULL_CSAUTH3_INSTANCE ▪ C 전용 함수에 pbInstance 에 NULL 이 입력됨 ▪ 코드를 확인하여 NULL 이 입력되지 않도록 수정
2907	▪ WARNING_INVALID_CSAUTH3_INSTANCE ▪ C 전용 함수의 pbInstance 에 유효하지 않은 메모리가 입력됨 ▪ CSAuth3_CreateInstance() 가 제공한 메모리를 입력되도록 수정
2908	▪ WARNING_CRUSHED_CSAUTH3_INSTANCE ▪ C 전용 함수의 pbInstance 에 유효하지 않은 메모리가 입력되어 크러쉬가 발생함 ▪ CSAuth3_CreateInstance 의 malloc 으로 생성된 메모리가 깨지거나 CSAuth3_Release 로 소거된 이후에 사용됨 ▪ CSAuth3_SetInstance / CSAuth3_GetInstanceSize 를 이용해서 자체 메모리 풀을 사용하는 방식으로 수정하는 것을 권장
2950	▪ WARNING_PACKET_MEM_SHORTAGE ▪ 정의된 패킷보다 큰 패킷의 생성을 시도함 ▪ 있을 수 없는 상황이나 해커, 혹은 CS 인증 버그에 의해서 발생할 수도 있음 게임가드 측에 문의

2990	▪ 프로토콜 업데이트시 발생할 수 있는 임시 에러로서 문제의 상황이 아닌 의도된 에러코드임
2999	▪ 게임가드 테스트 용도로 임의 발생되는 에러코드

- 3000 번대 에러 : CS 인증 개발 미스 - CS 인증 개발상의 문제로서 에러가 발생할 경우 게임가드에 문의 바랍니다. 보통 CS 인증 개발시에만 발생할 수 있는 에러로서 제공된 모듈에서는 발생할 가능성이 없습니다.

3000 3001 3002 3003 3010 3011	▪ CS 인증 기능 추가 중에 발생할 수 있는 개발자 에러 ▪ 사용중에는 발생할 가능성이 없으며 만약 발생 시 게임가드 측에 문의

- 3100 번대 에러 : 시스템 에러 - 시스템이 발생시키는 오류로서 메모리 생성 오류인 경우가 대부분 입니다.

3100 3101	▪ CS 인증을 통한 패턴차단 기능 사용을 위한 메모리 할당 실패 ▪ new 를 이용한 수십 kb 정도의 메모리 확보를 실패한 상황으로서 서버 리부팅등 적절한 대응 책이 필요
3102	▪ DataBase/GGScan/Pattern.db 파일을 read 속성으로 open 할 수 없음 ▪ 재실행 이후 해결 안되면 OS 의 권한문제 해결, 혹은 하드디스크 용량 확인
3103	▪ DataBase 내의 숫자폴더내의 파일을 open 할 수 없음 ▪ 재실행 이후 해결 안되면 OS 의 권한문제 해결, 혹은 하드디스크 용량 확인
3104	▪ DataBase/GGScan/Pattern.db 파일을 write 속성으로 open 할 수 없음 ▪ 재실행 이후 해결 안되면 OS 의 권한문제 해결, 혹은 하드디스크 용량 확인
3105	3104 와 동일

- 3200 번대 에러 : CS 인증 적용 미스 - CS 인증을 올바르지 않게 사용하여 발생하는 게임서버 개발자 미스에 의한 에러입니다.

3200	▪ ERROR_CANNOT_FIND_MODULE ▪ ggauth32_x 모듈을 찾을 수 없음 ▪ InitCSAuth3 함수에 입력하는 경로에 해당 파일을 제대로 복사하였는지 검토 ▪ 32비트 Server 에서 ggauth64_x 를 사용하거나 64비트 Server 에서 ggauth32_x 를 사용하는 경우 ▪ ggauth32_xx 에서 뒤의 숫자를 지운 경우 (숫자를 지우면 안 됨) ▪ InitCSAuth3() 에 "./xxx" 등으로 상대경로를 입력한 경우 CurrentDirectory 가 다른 경로를 가리키는 경우에 발생할 수 있음, 호출 전에 SetCurrentDirectory 로 올바른 경로를 재 설정해서 해결
3201	▪ ERROR_CANNOT_LOAD_MODULE ▪ ggauth32_x 모듈을 Load 하는데 실패함 ▪ 모듈의 Dependency 문제로 인해서 Load 되지 않는 상황으로 일반적으로 VS C++ 2008 재배포 패키지를 설치하면 해결 됨 ▪ 사용하는 모듈이 현재 OS 용이 맞는지 확인하고, 모든 것이 정상인 경우 게임가드 측에 문의
3202	▪ ERROR_INVALID_GLOBAL_FUNCTION ▪ 특정 함수를 모듈에서 import 하는데 실패함 ▪ InitCSAuth3() 함수를 호출 하지 않고 다른 함수를 호출 한 경우 ▪ 발생 가능성 낮음, 게임가드 측에 문의

3203	▪ ERROR_INVALID_OBJECT_FUNCTION ▪ 특정 함수를 모듈에서 import 하는데 실패함 ▪ InitCSAuth3() 함수를 호출 하지 않고 다른 함수를 호출 한 경우 ▪ 발생 가능성 낮음, 게임가드 측에 문의
3204	▪ ERROR_INVALID_CCSAUTH_DATA ▪ CCSAuth3::data 가 변조됨 ▪ CCSAuth3 객체의 메모리가 초기화되거나 변조됨
3210	▪ ERROR_CONFIG_PATH_CREATE_FAILED ▪ csauth3.cfg 로딩을 위한 경로생성 실패함 ▪ InitCSAuth3 에 입력된 경로값이 너무 긴 경우로서 경로를 줄여줌
3211	▪ ERROR_CONFIG_PATH_OPEN_FAILED ▪ csauth3.cfg 파일 Open 에 실패함 ▪ InitCSAuth3 에 입력된 경로에 csauth3.cfg 가 있는지 확인한다.
3212	▪ ERROR_INVALID_CONFIG_INFO ▪ csauth3.cfg 에서 추출한 값이 정상적인 데이터 범위를 넘음 ▪ 매뉴얼을 참조하여 유효한 값을 가지도록 csauth3.cfg 을 재설정 한다.
3213	▪ ERROR_NORMAL_LOG_OPEN_FAILED ▪ NormalLog 파일의 Create/Open 실패 ▪ NormalLog 폴더 생성, 파일 생성 권한 문제 해결 csauth3.cfg 의 NormalLog 의 입력 값이 유효한지 확인
3214	▪ ERROR_UPDATE_LOG_OPEN_FAILED ▪ UpdateLog 파일의 Create/Open 실패 ▪ UpdateLog 폴더 생성, 파일 생성 권한 문제 해결 csauth3.cfg 의 UpdateLog 의 입력 값이 유효한지 확인
3215	▪ ERROR_DEBUG_LOG_OPEN_FAILED ▪ DebugLog 파일의 Create/Open 실패 ▪ DebugLog 폴더 생성, 파일 생성 권한 문제 해결 csauth3.cfg 의 DebugLog 의 입력 값이 유효한지 확인
3216	▪ ERROR_READ_MIN_PROTOCOL_FAILED ▪ DataBase/MinProtocol.db 파일 Create/Open 실패 ▪ DataBase 폴더 생성, 파일 생성 권한 문제 해결 csauth3.cfg 의 DataBase 의 입력 값이 유효한지 확인
3220	▪ ERROR_INVALID_GGSCAN_DB_PATH ▪ DataBase/GGScanDB 경로정보가 유효하지 않음 ▪ csauth3.cfg 의 DataBase 의 입력 값이 유효한지 확인
3221	▪ ERROR_DIR_CREATION_FAILED ▪ csauth3.cfg 에 저장된 폴더들를 생성하는데 실패함 ▪ csauth3.cfg 의 폴더중에 생성이 불가능한 폴더 경로가 있지 않은지 확인 폴더 생성 권한 문제 해결
3230	▪ ERROR_INVALID_CALLBACK_TYPE ▪ SetCallbackFunction 의 nFunctionType 이 유효하지 않음 ▪ nFunctionType 의 입력값이 enum CallbackType 의 값인지 확인
3240	▪ ERROR_NO_CLIENT_FILES ▪ CheckClientType=1 이면서 ClientDirectory 폴더에 게임 클라이언트가 1개도 없는 경우

- 3300 번대 에러 : 서버 CS인증 에러코드 - 서버에서 발생한 정상적인 CS인증에 의한 유저 접속해제 에러코드

3300	▪ ERROR_GET_TWICE_WITHOUT_CHECK_AT_FIRST ▪ 유저가 첫인증에서 응답을 한번도 하지 못한 경우 Check 함수 호출 없이 Get 함수가 2회 연속 호출 되는 경우
3301	▪ ERROR_HIGHER_THAN_SUPPORT_PROTOCOL ▪ 유저의 프로토콜이 ggauth_x 모듈이 지원하는 프로토콜 보다 높은 경우 ▪ 많은 양이 발생하는 경우 낮은 버전의 ggauth_x 모듈을 최신 버전으로 교체
3302	▪ ERROR_LOWER_THAN_CURRENT_PROTOCOL ▪ 유저의 프로토콜이 서버 프로토콜 보다 낮은 경우 ▪ 많은 양이 발생하는 경우 지원부에 요청해서 게임가드의 프로토콜을 조정함
3303	▪ ERROR_LOWER_THAN_CURRENT_GGVERSION ▪ 유저의 내부버전이 서버 내부버전 보다 낮은 경우 ▪ 많은 양이 발생하는 경우 지원부에 요청해서 게임가드의 내부버전을 조정함
3304	▪ ERROR_INVALID_ORIGINAL_CRC ▪ 암호화 전의 원본 패킷이 수정 됨
3305	▪ ERROR_INVALID_PACKET_CRC ▪ 생성된 최종 패킷이 변경됨 ▪ 적용 초기에는 게임 개발사에서 패킷 전송구현 실수로 발생하였을 가능성이 큼
3306	▪ ERROR_INVALID_PACKET_HEADER ▪ 유효하지 않은 패킷 해더를 가지고 있음
3307	▪ ERROR_INVALID_PROTOCOL_NUMBER ▪ 유저의 프로토콜 번호가 0 임
3308	▪ ERROR_INSUFFICIENT_PACKET_BUFFER_SIZE ▪ Check 에 전달 받은 패킷 크기가 너무 작음
3309	▪ ERROR_GET_TWICE_WITHOUT_CHECK_AT_NOT_FIRST ▪ 유저가 첫인증이 아닌 상황에서 응답을 하지 않는 경우 Check 함수 호출 없이 Get 함수가 2회 연속 호출 되는 경우 클라이언트가 응답을 한번이상 하였으나 이후에 응답을 못한 상황으로 랙이나 네트워크등 다른 문제일 가능성이 있음
3310	▪ ERROR_TOO_BIG_PACKET ▪ 4096 보다 큰 사이즈의 패킷이 입력됨 ▪ CCSAuth3 의 메모리 관리가 잘못되어 쓰레기 값으로 채워지는 경우도 있음
3311	▪ ERROR_TOO_SMALL_PACKET ▪ 해킷헤더보다 작은 사이즈가 입력된 경우 ▪ 해커에 의한 악의적인 입력, 혹은 개발사 구현 실수
3312	▪ ERROR_PACKET_BUFFER_OVERFLOW ▪ 패킷 사이즈가 유효하지 않아 잘못된 메모리에 접근 되는 경우 ▪ 해커에 의한 악의적으로 패킷사이즈나 데이터가 조작된 경우
3313	▪ ERROR_EXCEPTION_OCCURED ▪ CSAuth3 의 객체가 유효하지 않은 상태에서 함수가 호출되는 경우 ▪ 서버의 구조적인 문제
3314	▪ ERROR_INVALID_PACKET_AT_FIRST_CHECK ▪ Check 함수에 입력된 Packet 이 올바르지 않음 ▪ 해커에 의한 변조된 패킷, 게임개발사 초기구현 미스

- 3400 번대 / 3500번대 에러 : 서버 CS인증 에러코드 - 서버에서 발생한 정상적인 CS 인증에 의한 유저접속해제 상황 중 내부 차단 기능에 의한 에러코드

3400	▪ 잘못된 데이터
3401 3402	▪ 암복호화 실패
3403	▪ 프로토콜 보정 미스
3404	▪ 입력받은 패킷 사이즈가 너무 큼
3410	▪ 필수 정보가 부족함

3411 3412 3413 3414 3415	
3420 3421	▪ 내부버전 문제
3430 3431 3432	▪ 유일값 문제
3440 3441 3442	▪ 게임가드 동작 여부 문제
3450	▪ CS 인증 알고리즘 문제 ▪ Get 에서 추출한 packet data 를 Check 에 그대로 입력하면 이 에러가 발생합니다. Check 함수는 클라이언트의 callback 함수에 전달된 packet data 를 서버로 전달하고 이를 packet 에 복사한 후에 호출 해주어야 합니다.
3460	▪ CS 인증 디버깅 문제
3470	▪ ERROR_CLIENT_NOT_ALLOW ▪ 수동 구버전 클라이언트 차단기능으로 구버전 클라이언트 차단
3471	▪ ERROR_CLIENT_AUTO_NOT_ALLOW ▪ 자동 구버전 클라이언트 차단기능으로 구버전 클라이언트 차단

- 3600 번대 에러 : 클라이언트 CS인증 에러코드 - 클라이언트에서 발생한 정상적인 CS 인증에 의한 유저접속해제 에러코드

3600	▪ ERROR_CLIENT_INVALID_PACKET_CRC ▪ 클라이언트에 생성된 최종 패킷이 변경 되어서 도착함 ▪ 적용 초기에는 게임 개발사에서 패킷 전송구현 실수로 발생하였을 가능성이 큼
3601	▪ ERROR_CLIENT_INVALID_ORIGINAL_CRC ▪ 클라이언트에 암호화 전의 원본 패킷이 변경 되어서 도착함
3602	▪ ERROR_CLIENT_INVALID_HEADER ▪ 클라이언트에 도착한 패킷의 헤더가 유효하지 않음
3603	▪ 암복호화 실패
3604	▪ 필수 정보가 부족함
3605	▪ ERROR_CLIENT_INVALID_PACKET_SIZE ▪ 너무 작은 패킷용량이 입력됨
3699	▪ 게임가드 테스트 시에 임의 발생되는 에러코드임

2장 게임가드 CS 인증 FAQ

1. 게임가드 FAQ

Q : CS 인증에서 직접 소켓통신을 이용하여 클라이언트로 패킷을 전송하나요?

A : 아닙니다. CS 인증은 단순히 패킷Data 만 생성/확인 할 뿐이고 이것을 클라이언트로 전송하고 클라이언트에서 서버로 수신하는 것은 게임개발사에서 직접 구현해 주어야 합니다. CS 인증은 네트워크에 전혀 관여하지 않습니다.

Q : packet 크기가 어느 정도 인가요?

A : 일반적인 경우 70~200 byte 정도 발생하지만, 핵툴 유저에게 발생되는 패킷이나 게임가드 사내 테스트 시에는 2000byte 까지도 발생하는 경우도 있습니다. 추후 기능이 추가된다면 크기가 더 증가될 가능성도 존재합니다.
가변적인 크기의 패킷을 생성하시는 것이 제일 좋으며, 크기를 고정한다면 4096byte 로 고정하시고 실제 사이즈만 올바르게 서버/클라이언트 간에 전달해 주시면 됩니다.

Q : 메모리 Leak 이 발생합니다.

A : 서버인증 3.0은 heap 을 전혀 사용하지 않기 때문에 자체적인 메모리 Leak 이 발생할 수 없는 구조 입니다. CCSAuth3 객체를 관리하는 부분을 검토해주시기 바랍니다. CSAuth3_ 으로 시작하는 C 전용 함수를 사용하는 경우에는 malloc 을 호출하므로 적용 방식에 따라 릭이 발생할 수도 있습니다. CSAuth3_Release() 가 반드시 호출되도록 구현해 주시는방법 또는 자체적인 메모리 풀을 사용하는 방식을 사용해 주시기 바랍니다.

2. 고객지원

㈜잉카인터넷에서는 nProtect GameGuard 사용자에게 E-Mail, 전화를 통해 고객 지원을 하고 있습니다.

게임가드와 관련된 에러는 메시지와 에러코드를 사용자에게 보여주면 고객 지원시 도움이 됩니다.
자주 발생하고 해결 방법이 명확한 문제는 게임에서 직접 해결법을 메시지로 보여주면 좋습니다.
바이러스나 스파이웨어에 의해 실행이 제대로 되지 않는 경우가 많습니다. 사용자들에게 최신 백신과 스파이웨어 검사 프로그램을 사용하여 PC를 검사해보도록 권합니다.
그래도 해결되지 않는 문제는 문제가 발생한 직후에 게임 폴더 안의 GameGuard 폴더에 있는 *.erl 파일들을 (국내: game1@inca.co.kr 해외: game2@inca.co.kr) 로 첨부하여 메일 주시면 분석 후 해결하도록 하겠습니다.

게임폴더 위치 찾기: 게임 아이콘 바로가기->오른쪽 버튼 클릭->속성(등록정보) 클릭->대상 찾기 클릭

- nProtect GameGuard 사용시 문의 사항이나 문제점이 있으신 경우 언제든지 아래와 같은 방법으로 ㈜잉카인터넷으로 연락 주시기 바랍니다.

① 게임실행이 안 되거나 오류 발생 시점에서 게임을 종료합니다.

② 게임실행이 안되는 상황 설명과 함께 게임 폴더 안의 GameGuard 폴더에 있는 *.erl 파일들을 첨부해서 메일 주시면 됩니다.

③ 이메일 접수
일반유저 : (국내: game1@inca.co.kr 해외: game2@inca.co.kr) 을 통해 문의사항 접수
고 객 사 : GameService@inca.co.kr 을 통해 문의사항 접수

게임가드 서버이증에 대한 오류 발생시 아래와 같이 진행을 부탁드립니다.

1. 서버인증 오류 발생시 오류내용에 대한 사항을 메일로 문의해 주시기 바랍니다.

2. 서버인증 오류에 대한 서버로그를 ERL 로그와함께 회신주시기 바랍니다.
- 오류 발생시점에서의 Erl 로그와 동일시간대의 서버 오류로그를 취합해 주시면 됩니다.

■ (주)잉카인터넷 게임보안 사업본부 글로벌사업부 Feedback

메일: gameservice@inca.co.kr www.gameguard.kr

서울시 구로구 구로3동 235-2 에이스 하이엔드타워 12층 1204호 (우 152-848)

Security, For a More Joyful Gameplay.

MEMO