nProtect GameGuard

CS Authentication Manual

■ (주)잉카인터넷 게임보안 사업본부 개발부 엔진개발팀

DV-101215

Ver. 1.0

업체 제공용

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

차 례

1장 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. CS인증 2.5는 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.

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

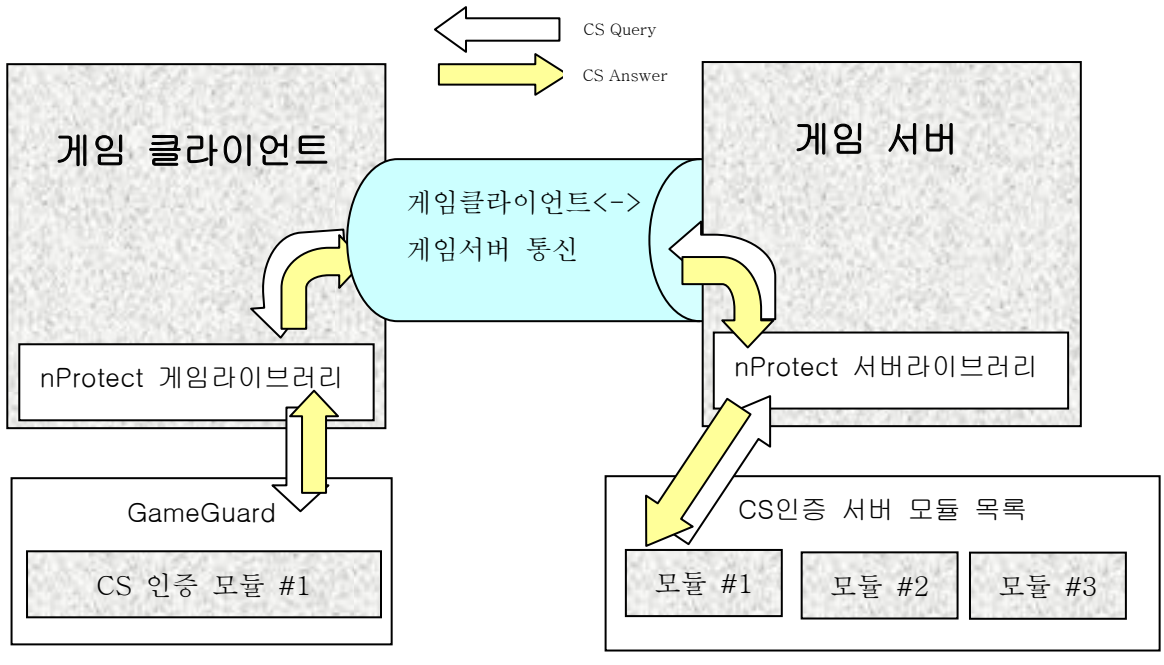
서버에서는 항상 최신 버전의 GameGuard 의 접속만을 허용함으로 새로운 해킹 방법이 배포되더라도 GameGuard 의 버전을 올려서 업데이트 함으로써 즉각적인 대응이 가능하며 구 버전 인증으로의 접속을 제한하게 됩니다.

- 다중모듈 로딩

CS인증은 인증 모듈의 해킹에 대비하여 한꺼번에 수개씩 모듈을 로딩하여 모듈이 해킹 되더라도 게임서버의 조작 없이 게임가드만의 업데이트로 인증 모듈을 변경시킴으로써 즉각적인 대응이 가능합니다.

2장 시스템 구성

1. 시스템 구성도



nProtect GameGuard CS Authendication

2. 구성 모듈

- 서버에 적용할 파일들로만 구성됨. 클라이언트는 NPGameLib.lib 에 의해 제공됨

[헤더파일]

- ggsrv25.h - CS인증 라이브러리 서버 헤더 파일
- ggsrv25_cpp_c.h - CS인증 라이브러리 서버 헤더 파일 (C 전용 헤더 파일)

[라이브러리]

- ggsrvlib25_MD.lib - CS인증 서버 라이브러리 파일 (Win32 or Win64 - Multithreaded DLL)
- ggsrvlib25_MT.lib - CS인증 서버 라이브러리 파일 (Win32 or Win64 - Multithreaded)
- libggsrv25.a - CS인증 서버 라이브러리 파일 (Linux / Solaris8, 10 / FreeBSD 4.11)

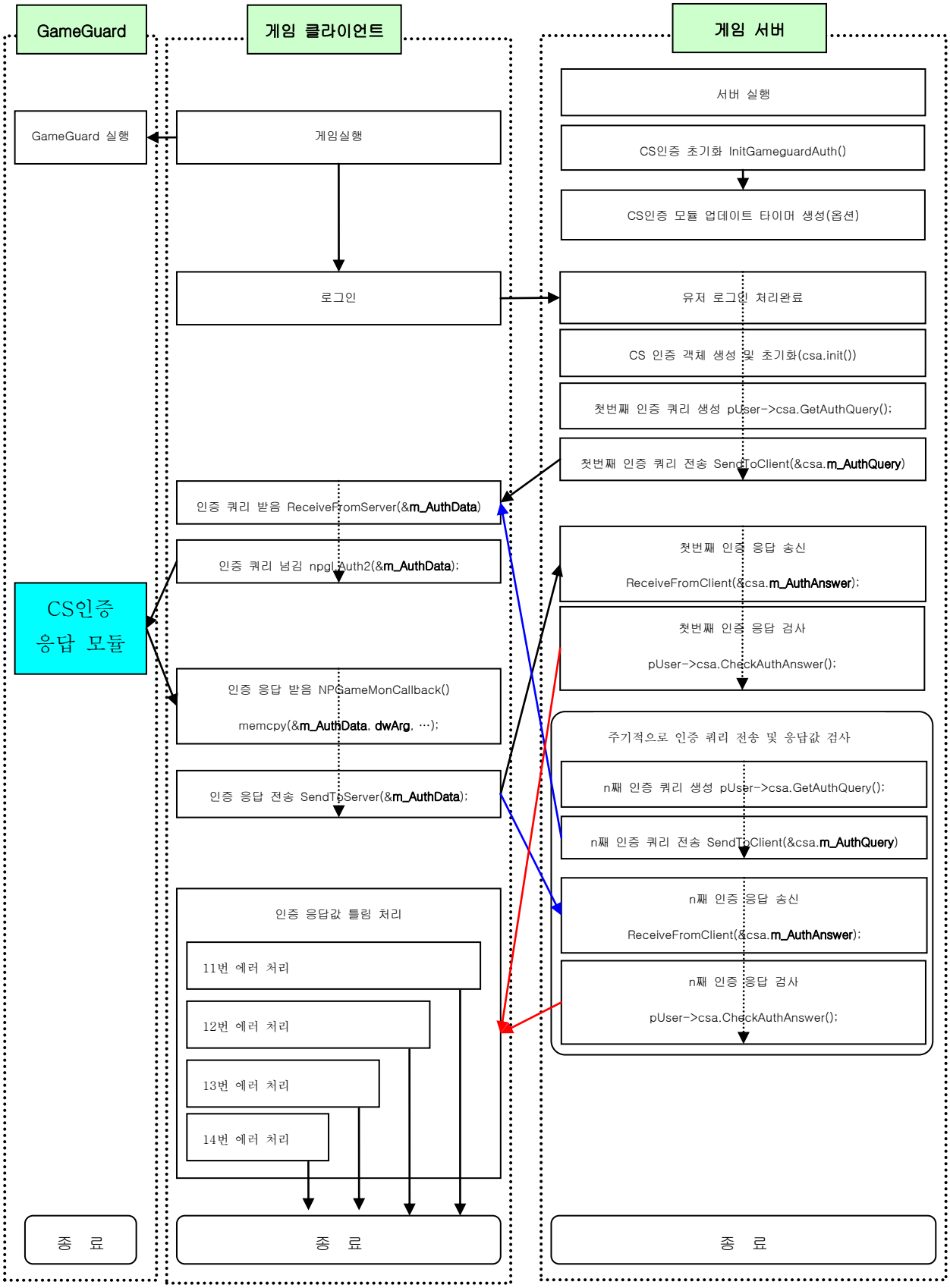
[CS인증 모듈]

- ggauth##.dll - CS인증 서버 알고리즘 모듈 ## 번 (Win32 or Win64)
- libggauth2.so.## - CS인증 서버 알고리즘 모듈 ## 번 (Linux / Solaris8,10 / FreeBSD)

[기타]

- csauth2.cfg - CS인증 서버 알고리즘 모듈 로딩 목록 리스트

3. 프로세스


GameGuard
게임 클라이언트
게임 서버
서버 실행
GameGuard 실행
게임실행
CS인증 초기화 InitGameguardAuth()
CS인증 모듈 업데이트 타이머 생성(옵션)
로그인
유저 로그인 처리완료
CS 인증 객체 생성 및 초기화(csa.init())
첫번째 인증 쿼리 생성 pUser->csa.GetAuthQuery();
첫번째 인증 쿼리 전송 SendToClient(&csa.m_AuthQuery)
인증 쿼리 받음 ReceiveFromServer(&m_AuthData)
인증 쿼리 넘김 npgl_Auth2(&m_AuthData);
첫번째 인증 응답 송신
ReceiveFromClient(&csa.m_AuthAnswer);
첫번째 인증 응답 검사
pUser->csa.CheckAuthAnswer();
CS인증
응답 모듈
인증 응답 받음 NPGameMonCallback()
memcpy(&m_AuthData, dwArg, …);
주기적으로 인증 쿼리 전송 및 응답값 검사
n째 인증 쿼리 생성 pUser->csa.GetAuthQuery();
인증 응답 전송 SendToServer(&m_AuthData);
n째 인증 쿼리 전송 SendToClient(&csa.m_AuthQuery)
n째 인증 응답 송신
ReceiveFromClient(&csa.m_AuthAnswer);
인증 응답값 틀림 처리
n째 인증 응답 검사
pUser->csa.CheckAuthAnswer();
11번 에러 처리
12번 에러 처리
13번 에러 처리
14번 에러 처리
종 료
종 료
종 료

3장 시스템 적용

1. 준비 사항

- 제공해드린 헤더파일(ggsrv25.h)을 적당한 곳에 위치 시키십시오.
- 제공해드린 라이브러리 파일을 적당한 곳에 위치시키고 프로젝트 링크 옵션에 해당 라이브러리 파일을 추가 하십시오.
  - 윈도우 계열 서버환경(32 / 64 비트)의 경우 라이브러리 파일은 **ggsrvlib25.lib** 입니다.
  - Multithreaded (MT) / Multithreaded DLL (MD) 중 맞는 라이브러리를 사용하면 됩니다.
  - 리눅스 / 솔라리스 / FreeBSD 서버환경의 경우 라이브러리 파일은 **libggsrv25.a** 입니다. 컴파일러 버전에 맞게 사용하십시오.
- 제공해드린 CS 인증 모듈을 게임서버 실행위치의 하위에 적당히 위치시키십시오.
  - csauth2.cfg 와 인증모듈은 항상 같은 위치에 존재해야 합니다.
  - 윈도우 계열의 서버환경에서는 ggauthXX.dll 이며, 리눅스 / 솔라리스 / FreeBSD 서버환경에서는 libggauth2.so.XX 입니다.
- 설명서는 C++ 개발 환경을 기준으로 작성되었습니다. 서버가 C 으로만 구현되어 있을 경우 C 전용 적용을 같이 참조하십시오.
- 게임서버에 해당 모듈을 적용하기 전에 2 장 3 절의 “프로세스 구성”을 참고하면 적용에 더욱 도움이 됩니다.
- CS 인증 라이브러리의 함수 전달인자는 “**[참고#1] 함수 프로토 타입**” 을 참고 하십시오.

2. 서버 적용

가. CS인증 관련 함수를 사용하는 모든 소스파일에서 함수의 선언을 참조할 수 있도록 ggsrv25.h 를 인클루드(include) 를 하십시오.

```
ex) cat tserver.cpp
#include <stdio.h>
…
#include "/nProtect/ggsrv25.h"
```

```
ex) cat tserver.cpp
int main(void)
{…
int main(void)
{
}
… ( 초기화 완료 ) …
        dwRet = InitGameguardAuth("./nProtect/", 50, true, 0x03);
        if( dwRet != ERROR_SUCCESS ){
                printf("Fail[%d] InitGameguardAuth()₩n", dwRet);
                exit(1);
        }
        SetUpdateCondition(30, 50); // 30분동안 50% 이상
        CreateTimer(NPROTECT_UPDATE_TIMER, 5 * 60 * 1000); //추가적용사항.
}
```

나. 게임서버의 초기화가 완료된 후에 CS인증을 초기화 시키십시오. CS인증을 초기화 하여 리턴값이 성공(ERROR_SUCCESS = 0) 하기 전에 인증 쿼리를 생성하거나 응답 검사를 하시면 안됩니다. SetUpdateCondition 은 해당함수 설명을 참조.

다. 접속유저를 구분하는 구조체나 객체에 CCSAuth2 객체를 참조하는 변수를 추가합니다. 모든 접속유저 객체는 CCSAuth2 객체를 각각 생성해야 합니다. (CCSAuth2 객체는 CS인증 정보를 유저별로 각각 저장합니다.)

```
ex) cat User.cpp
class CUser {
…
public:
CCSAuth2* csa;
}

ex) cat tserver.cpp
…
int main(void)
{
//사용자 객체에 CSAuth2 객체 변수(csa) 추가 및 메모리 할당
CreateUser(&pUser);
pUser.csa = new CCSAuth2();
pUser.csa.Init(); // pUser 객체를 Pool 에 미리 생성하여 관리할 경우 다시 사용할 때 csa 의 초기화 함수를 호출해 줍니다.
}
```

라. 일정한 주기마다 CS인증 쿼리를 생성하여 클라이언트에게 전송합니다. "**[참고#2]인증주기**" 를 참조하여 보안성 강화처리를 꼭 해주십시오. 에러로그 작성법은 [참고#4]를 참조하십시오.

```
ex) cat tserver.cpp
…
int main(void)
{
… ( 인증 타이머(각 사용자별로 따로 처리할 경우 랙이 발생하지 않음) )
GG_AUTH_DATA ggData;
dwRet = pUser->csa.GetAuthQuery();
if(dwRet != ERROR_SUCCESS)
{
//에러처리
}
memcpy(&ggData, &pUser->csa.m_AuthQuery, sizeof(GG_AUTH_DATA));
sendToClient(&ggData, sizeof(ggData))
}
```

마. 클라이언트에서 송신한 패킷 중에 CS인증 패킷을 받을 경우 각 유저별로 송신한 응답값에 대한 인증을 검사합니다. 에러로그 작성법은 [참고#4]를 참조하십시오.

```
ex) cat tserver.cpp
…
int main(void)
{
… ( 사용자로부터 인증 값을 받음 )
GG_AUTH_DATA ggData;
recvFromClient (pUser, ggData, sizeof(GG_AUTH_DATA));
memcpy(&pUser->csa.m_AuthAnswer, &ggData, sizeof(GG_AUTH_DATA));
dwRet = pUser->csa.CheckAuthAnswer ();
if(dwRet != ERROR_SUCCESS)
{
//에러처리
}
}
```

바. 사용자의 접속이 해제되었을 경우 CSAuth2 객체를 소멸(delete)시키거나 Close() 를 호출합니다.

```
ex) cat tserver.cpp
…
int main(void)
{
… ( 접속 종료 )
//pUser 를 삭제할 경우
delete pUser->csa;
//pUser 를 pool 에 반환할 경우
pUser->csa.Close();
LogOut(pUser);
}
```

사. 서버를 종료시킬 때에는 CS인증 모듈을 안전하게 해제할 수 있도록 합니다.

```
ex) cat tserver.cpp
…
int main(void)
{
… ( 서버 종료 )
CleanupGameguardAuth();
}
```

아. CS인증 라이브러리에서는 ERROR 레벨의 로그와 DEBUG 레벨의 로그를 생성하고 이 로그를 NpLog() 라는 함수에 전달하게 됩니다. NpLog() 는 선언만 되어 있을 뿐 이 함수에 대한 구현은 각 서버프로그램에서 구현해야 하며 전달인자로 넘겨받은 로그(문자열)를 해당 서버시스템에 맞게 남기셔야 합니다. **NpLog() 를 구현하지 않는다면 링크에서 에러가 발생하게 됩니다.** DEBUG 레벨의 로그는 그 양이 많으므로 서버에 부하를 일으킬 수도 있습니다. 일반적인 상황에는 ERROR 로그를 문제가 발생해서 보다 자세한 로그를 남겨야 할 시에는 DEBUG 와 ERROR 로그 전부를 남기도록

하시면 됩니다.

```
GGAUTHS_API void NpLog(int mode, char* msg){
        //각 게임사의 로그 작성 정책에 맞게 로그를 남깁니다.
        if(mode & (NPLOG_DEBUG | NPLOG_ERROR)) //Select log mode.
                printf(msg);
};
```

자. CS인증 라이브러리에서는 CS인증 버전 및 프로토콜이 업데이트 되었을 때 GGAuthUpdateCallback 함수를 통해 보고를 받게 됩니다. 이것 또한 NpLog 함수와 마찬가지로 라이브러리에서 호출해주므로 서버소스 내에 구현되어 있어야 링크에서 에러가 발생하지 않습니다. GGAuthUpdateCallback 함수에 대한 처리는 보통 REPORT를 서버 로그에 남깁니다.

```
GGAUTHS_API void GGAuthUpdateCallback(PGG_UPREPORT report){
        printf("GGAuth version update [%s] : [%ld] -> [%ld] ₩n",
                report->nType==1?"GameGuard Ver":"Protocol Num",
                report->dwBefore, report->dwNext);
};
```

3. 클라이언트 적용

가. 서버에서 CS인증 2.x (2.0, 2.5 포함) 을 전송 받았을 때 게임가드에게 바로 전달합니다.

```
GG_AUTH_DATA m_AuthData;
// 서버로부터 인증 패킷을 수신
ReceiveFromServer(&m_AuthData);
npgl.Auth2(&m_AuthData); // 게임가드로 전달
```

나. 게임가드의 Callback 함수인 NPGameMonCallback 에서 CS인증 2.x 응답 값을 받으면 그대로 서버에 전송합니다.

```
// GameGuard 콜백 함수
bool CALLBACK NPGameMonCallback(DWORD dwMsg, DWORD dwArg){
        switch (dwMsg)
        {
        case NPGAMEMON_CHECK_CSAUTH2:
                memcpy(&m_AuthData, (PVOID)dwArg, sizeof(GG_AUTH_DATA));
                // 서버로 인증 패킷을 전송. 이때 Callback 함수는 게임가드 프로세서가
호출하므로 동기화가 필수
                SendToServer(&m_AuthData);
                return true;
        }
}
```

4. 추가(Option) 적용

- 위 2, 3 의 적용방법은 CS인증을 처리하기 위한 가장 기본적인 적용 방법입니다. 각 게임의 게임시스템 및 서버 환경이 다양한 만큼 추가 적용에서 필요한 부분을 꼭 적용시키십시오.

가. CS인증 모듈은 내부에서 따로 타이머를 생성하여 CS인증 알고리즘 번호를 자동으로 변경 시킵니다. CS인증 알고리즘이 변경됨으로써 하나의 계산 알고리즘이 해커에 의해서 분석되더라도 일정시간(각 모듈마다 틀리지만 보통 3시간)이 지나면 분석된 알고리즘은 사용할 수 없게 됩니다. 하지만 CS인증 모듈 내부에서 생성하는 타이머가 시스템의 특성에 이해서 동작하지 않을 수도 있으므로 게임서버에서 직접 타이머를 생성하여 알고리즘 번호 변경 함수를 호출 하도록 권장합니다.

A. CS인증 모듈 내부에서 타이머를 생성하지 않도록 InitGameguardAuth() 함수에서 3번째 전달인자를 true 로 지정합니다.

```
dwRet = InitGameguardAuth("./nProtect/", 50, true, 0x03);
```

B. 게임서버에서 직접 타이머를 하나 생성합니다.

```
#define NPROTECT_UPDATE_TIMER 1234   //타이머 ID
CreateTimer(NPROTECT_UPDATE_TIMER, 180 * 60 * 1000); //3시간 주기 권장
```

C. 타이머가 호출될 때 마다 GGAuthUpdateTimer() 를 호출합니다.

```
void PrtcOnTimer(int timerId) {
        if(timerId == NPROTECT_UPDATE_TIMER)
                GGAuthUpdateTimer();
}
```

나. CS인증 2.5 는 서버에 최신 게임가드의 CS인증 모듈 번호를 항상 유지합니다. 현재 서버에 적용된 모듈 정보를 알고 싶을 때 ModuleInfo() 함수를 통해서 확인 할 수 있습니다. ModuleInfo 함수는 현재 로딩된 모듈에 대한 정보를 문자열로 표시합니다.

```
char msgBuf[512];
int ret = ModuleInfo(msgBuf, 512);
if(ret != NPGG_INFO_SUCCESS){
//에러처리
}
```

(로그 샘플)

```
PRTC[0x00010051]C[0x00d5e9c0]G[0x00d5e8a0]E[0x00d5ea90]D[0x00d5ea90]U[0x00d5e890]:    state[ACTIVE]    ==>
PRTC[0x00010050]C[0x00d8fc90]G[0x00d8fb70]E[0x00d8fd60]D[0x00d8fd60]U[0x00d8fb60]    :    state[DISUSE]    ==>
PRTC[0x00010052]C[0x00dcfbe0]G[0x00dcfac0]E[0x00dcfcb0]D[0x00dcfcb0]U[0x00dcfab0]    :    state[STANDBY]    ==>
PRTC[0x00010053]C[0x00e23930]G[0x00e23800]E[0x00e23a00]D[0x00e23ac0]U[0x00e237f0]    :    state[STANDBY]    ==>
PRTC[0x00010054]C[0x00e6b5b0]G[0x00e6b480]E[0x00e6b680]D[0x00e6b740]U[0x00e6b470]    :    state[STANDBY]    ==>
PRTC[0x00010055]C[0x00ea8060]G[0x00ea7f40]E[0x00ea8130]D[0x00ea81f0]U[0x00ea7f30] : state[STANDBY] ==>
```

50~55번까지의 모듈을 로딩하였으며 현재 활성화(ACTIVE) 된 모듈은 51번입니다. C,G,E,D,U 항목은 CS인증 모듈의 export 함수 주소를 나타냅니다. state[ACTIVE] 는 현재 활성화된 모듈을 뜻하며 로딩된 모듈 중에 유일하게 하나만 존재합니다. state[STANDBY] 는 사용 예정된 모듈이며 state[DISUSE] 는 폐기된 모듈을 뜻합니다.

다. CS인증 2.5 는 각 사용자 별로 게임가드 인증모듈에 대한 정보를 Info 함수를 통해 표시할 수 있습니다. 해당 사용자에 대한 인증모듈에 대한 정보를 확인하고 싶을 때에는 Info 함수를 호출하십시오.

```
char msgBuf[512];
int ret = pUser->csa. Info(msgBuf, 512);
if(ret != NPGG_INFO_SUCCESS){
//에러처리
}
```

정보는 ModuleInfo 와 같은 형태입니다.

라. 현재 사용자가 어떤 버전의 게임가드를 사용하고 있는지 확인 할 수 있습니다.

```
int rtn = pUser->csa.CheckUpdated();
switch(rtn){
case NPGG_CHECKUPDATED_NOTREADY; //첫 번째 인증을 하지 않은 상태
case NPGG_CHECKUPDATED_LOW; //현재 서버의 버전보다 낮음.
case NPGG_CHECKUPDATED_HIGH; //현재 서버의 버전보다 높음
case NPGG_CHECKUPDATED_VERIFIED; //현재 서버의 버전과 같음.
}
```

마. CS인증 2.5 는 게임서버의 시작시 한꺼번에 여러 CS인증 모듈을 로딩하여 사용할 수 있습니다. 그리고 새로운 모듈을 서버가 동작하는 실시간으로 AddAuthProtocol() 함수를 통해서 추가할 수 있습니다. 하지만 실시간으로 모듈을 추가하는 것을 권장하지는 않습니다.

```
dwGGErrCode = AddAuthProtocol(sDllName);
if( dwGGErrCode != ERROR_SUCCESS )
{
// 에러 처리
}
```

바. 실시간 버전 업데이트를 기능으로 항상 최신 버전의 게임가드를 유지합니다. 다음 함수는 게임가드 버전 업데이트의 조건을 수동으로 설정합니다. 기본적인 수치는 30, 50 입니다. 게임가드 버전 업데이트에 대한 설명은 참고#5 를 참조하십시오.

```
SetUpdateCondition(30, 50); //30분 동안 버전 통계치의 50% 이상 버전을 새버전으로 확정합니다.
```

사. 실시간 버전 업데이트를 기능으로 항상 최신 버전의 게임가드를 유지합니다. 게임가드를 신버전으로 업데이트 할 경우 서버에서는 업데이트 조건을 검사하여 새 버전으로 업데이트를 하지만 아래 함수를 통해서 업데이트 상한 버전을 강제로 설정할 수 있습니다. 전달인자의 범위는 2003010101 ~ 2100010101 입니다. 이렇게 설정된 버전은 csauth2.cfg 파일과 동일한 경로에 csver.cfg 파일로 남겨지며 게임서버가 기동될 때 로딩이 되어서 적용이 됩니다. 주의) 버전 상한치를 제한할 경우 게임가드의 업데이트를 주의해야 합니다. 버전 상한치 제한 잉카와 긴밀한 협조로 진행되어야 합니다.

```
SetGGVerLimit(2007010101);  //게임가드 CS인증 버전을 2007010101 으로 제한합니다. (2007010101 를 포함한 그 이하의 클라이언트는 Error 발생 됨)
```

5. C 전용 적용

- C 로 구현된 서버 시스템의 경우 CSAuth2 클래스를 사용할 수 없습니다. 하지만 제공된 ggsrv25_cpp_c.h 에는 C 로 구현된 서버 시스템에 CS인증 라이브러리를 적용시킬 수 있도록 함수를 제공하고 있습니다. 위 "4. 서버 적용" 에서 csa 객체에 접근하는 부분만 아래와 같이 바꾸십시오.

가. 인클루드 하는 헤더파일을 ggsrv25.h 에서 ggsrv25_cpp_c.h 로 교체 합니다.

```
//#include "/nProtect/ggsrv25.h"
#include "/nProtect/ggsrv25_cpp_c.h"
```

나. 접속유저를 구분하는 구조체나 객체에 CSAuth2 객체 대신 LPGGAUTH 를 참조하는 변수를 추가합니다. 모든 접속유저 객체는 LPGGAUTH 변수를 각각 생성하고 초기화를 해야 합니다. (LPGGAUTH 객체는 CS인증 정보를 유저별로 각각 저장합니다.)

```
ex) cat User.cpp
class CUser {
…
public:
LPGGAUTH csa;
}

ex) cat tserver.cpp
…
int main(void)
{
CreateUser(&pUser);
pUser.csa = GGAuthCreateUser();
GGAuthInitUser(pUser.csa); // pUser 객체를 Pool 에 미리 생성하여 관리할 경우 다시 사용할 때 csa 의 초기화 함수를 호출해 줍니다.
}
```

다. 일정한 주기마다 CS인증 쿼리를 생성하여 클라이언트에게 전송합니다. "**[참조#2]인증주기**" 를 참조하여 보안성 강화처리를 꼭 해주십시오.

```
ex) cat tserver.cpp
…
int main(void)
{
… ( 인증 타이머(각 사용자별로 따로 처리할 경우 랙이 발생하지 않음) )
GG_AUTH_DATA ggData;
dwRet = GGAuthGetQuery(pUser->csa, &ggData);
if(dwRet != ERROR_SUCCESS)
{
//에러처리
}
sendToClient(&ggData, sizeof(ggData))
}
```

라. 클라이언트에서 송신한 패킷 중에 CS인증 패킷을 받을 경우 각 유저별로 송신한 응답값에 대한 인증을 검사합니다.

```
ex) cat tserver.cpp
…
```

```
int main(void)
{
… ( 사용자로부터 인증 값을 받음 )
GG_AUTH_DATA ggData;
recvFromClient (pUser, ggData, sizeof(GG_AUTH_DATA));
dwRet = GGAuthCheckAnswer(pUser->csa, &ggData);
if(dwRet != ERROR_SUCCESS)
{
//에러처리
}
}
```

마. 사용자의 접속이 해제되었을 경우 CSAuth2 객체를 소멸(delete)시키거나 Close() 를 호출합니다.

```
ex) cat tserver.cpp
…
int main(void)
{
… ( 접속 종료 )
//pUser 를 pool 에 반환할 경우
GGAuthCloseUser(pUser->csa);
//pUser 를 삭제할 경우
GGAuthDeleteUser(pUser->csa);
LogOut(pUser);
}
```

6. Linux / Solaris / FreeBSD 환경에서 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.
- CS인증 라이브러리는 Dynamically Loaded (DL) 라이브러리를 사용하므로, 컴파일 시에 -ldl 옵션을 사용하셔야 합니다.
- C 컴파일러를 사용하는 경우, C++ 관련 링크 에러가 나면 -lstdc++ 옵션을 사용하셔야 합니다.
- 동기화를 처리하기 위해 -pthread 옵션을 사용하셔야 합니다.

7. Win64 환경에서 적용

- Win32 환경과 차이점이 없습니다. 제공해드린 64비트 전용 라이브러리를 그대로 사용하시면 됩니다. (배포압축파일 형태 : CSAuth25_Win64_Prtc50_52_[xxxxxxxx].zip)

4장 주의 사항.

1. CS인증 기능 동작 On/Off 처리
   - CS인증의 실패는 서버에서 해당 사용자의 접속을 해제 하거나 로그를 남기는 등 게임 시스템에 큰 영향을 끼칩니다. CS인증 모듈의 관리를 잘못하여 여러 인증 모듈 중 하나의 인증모듈에 문제가 생기거나 게임가드의 업데이트 실수로 잘못된 버전을 업데이트 할 경우 서버에서는 대량의 CS인증 오류가 발생할 수도 있습니다. 예상하지 못한 사고에 최대한 빠르게 대응하기 위해서는 CS인증 기능을 긴급하게 해제(On/Off) 시킬 수 있도록 실시간 명령 옵션처리 하는 것을 권장합니다. 게임가드는 CS인증 패킷이 올 경우에만 반응하도록 수동적으로 동작하므로 서버에서 인증 패킷을 전송 하지 않는다면 게임가드는 CS인증을 하지 않으므로 안전하게 CS인증 기능을 OFF 시킬 수 있습니다. (다만 CS인증으로 막을 수 있는 핵이 사용될 수 있습니다.)
2. csa 객체의 재사용
   - csa 의 객체를 재사용 할 경우 항상 Init()후에 사용해야 하며 사용 완료 후 Close() 로 종료처리를 합니다.
3. 버전검사 실패 오류에 대한 처리(오류코드 12, 14번)
   - CS인증은 버전 확인을 위해서 버전 확인 쿼리를 생성하여 가장 처음에 주고 받습니다. 이때 12번, 14번 오류가 발생할 수 있으며 각각은 CS인증 모듈의 버전 틀림과 게임가드 버전 틀림입니다. 서버에서는 이런 오류를 발생시키는 클라이언트를 재시작 할 수 있도록 처리해야 합니다. CS인증 2.5는 항상 버전을 최신으로 유지합니다. 만일 게임가드의 버전을 업데이트 했을 때 업데이트 이전에 게임을 실행시킨 사용자가 서버에 접속을 못하여 혼란을 겪을 수 있으며 서버시스템에 따라서 다른 서버로의 이동이나 로그아웃 후에 다시 로그인이 되지 않을 수도 있습니다. 이런 경우 클라이언트의 재시작을 통한 게임가드의 업데이트 이외에는 방법이 없으므로 서버에서 12, 14번 오류가 발생했을 경우 클라이언트의 재시작을 꼭 유도해 주십시오.
4. 서버에서 CS인증 주기를 일괄적으로 처리할 경우
   - 인증주기 시간 직전에 로그인 한 사용자(로그인 하자마자 인증쿼리 생성 후 전송)의 경우 응답이 오기 전에 다시 인증쿼리를 생성(일괄적으로 쿼리를 생성하는 타이머에 의해)함으로써 응답 없음 에러가 발생할 수 있습니다. 이런 경우를 대비하여 인증쿼리를 전송한 시간을 저장하여 다음 인증 쿼리를 생성할 때 시간을 검사해서 정상적 사용자의 인증오류가 발생하지 않도록 처리해야 합니다.
   - 만일 인증주기를 5분으로 지정할 경우 타이머를 1분단위로 지정하고 각 사용자마다 마지막 쿼리 전송시간을 저장하고 인증 타이머(1분마다)에서 해당 사용자가 마지막 전송시간이 5분을 경과한 사용자만 인증 쿼리를 생성해서 전송 처리 합니다.
5. 서버에서 사용하는 모듈(서버라이브러리 및 DLL, SO파일들)은 외부 유출을 절대 금합니다.
   (실수라도 클라이언트 패키지에 포함되지 않도록 주의하시기 바랍니다.)

5장 테스트

1. 기본 테스트 (필수)

- CS인증 모듈의 정상적인 로딩과 인증 처리에 대한 기본 정상 동작 여부를 확인 합니다. 매뉴얼에서 설명한대로 적용이 완료되었을 경우 아래 절차에 따라서 기능 테스트를 하십시오.
- 보다 원활한 테스트를 위해서 InitgameguardAuth() 의 dwNumActive 의 전달인자를 10 정도로 조정해주시고 모든 로그를 남기도록 옵션을 조정해 주십시오. 또한 SetUpdateCondition() 의 nTimeLimit 를 5분, nCondition 을 50% 로 설정하셔서 버전 업데이트검사를 짧게 지정해주십시오.

가. 서버 실행 및 모듈 로딩 테스트

- 서버가 CS인증 모듈을 정상적으로 로딩하고 타이머가 제대로 동작하는지 확인 합니다.

A. InitGameguardAuth() 함수의 4번째 로그 플래그 인자에 NPLOG_DEBUG | NPLOG_ERROR 옵션으로 모든 로그를 작성하도록 설정합니다. (Real Server 에 올리기 전에는 계속 이 옵션을 유지하시는 것이 좋습니다.)

B. CS인증 모듈(*.dll or *.so , csauth2.cfg) 의 위치를 확인한 후에 서버를 실행시킵니다.

C. 모듈 로딩정보를 포함한 "**[C$][S lib] InitGameguardAuth() COMPLETED!**" 메시지가 표시되었다면 로딩이 완료된 것입니다.

D. InitGameguardAuth() 함수의 3번째 서버타이머 사용여부 인자를 true(권장) 으로 설정하였을 경우 서버 타이머가 호출될 때 마다 "**[C$][S lib] UpdateTimer Call [100##]**" 가 로딩된 모듈의 개수만큼 작성되며 타이머 기능이 정상 동작 하는 것입니다.

E. SetGGVerLimit() 함수를 호출했을 경우 csver.cfg 파일이 생성되며 이때 "**[C$][S lib] SUCCESS LOAD CFG : csver.cfg**" 란 메시지가 표시됩니다. 해당 파일이 존재하지 않을 경우 버전 상한치 제한기능이 OFF(기본값) 되며 "**[C$][S lib] Fail to load GGVer Limit at file(csver.cfg). Disabled GGVer limitation.**" 메시지가 표시 됩니다.

* 이 로그는 에러 발생을 의미하는 것이 아닙니다.

나. CS인증 패킷 송수신 테스트

- 서버에서 생성한 인증쿼리가 게임가드에게 정상적으로 전달되는지, 게임가드의 인증응답이 서버가 받아서 정상적으로 검사를 하는지 확인합니다.

A. **가.** 절차를 모두 마친 후에 잉카에서는 서버에 적용된 CS인증 모듈에 대응되는 게임가드 모듈을 준비하여 업체에 제공합니다.

i. GameGuard 를 처음 적용하는 경우 - CS인증 2.5 라이브러리와 게임가드 모듈을 함께 제공하며 게임가드 업데이트 서버에 제공해드린 CS인증 2.5 모듈에 대응되는 게임가드로 구성합니다.

ii. CS인증 2.0 에서 2.5 로 업그레이드 하는 경우

1. 비공개 테스트 서버 전용 게임가드 업데이트 서버가 존재할 경우
   - 테스트 일정에 맞추어서 CS인증 2.5 모듈에 대응되는 게임가드를 비공개 테스트서버용 게임가드 업데이트 서버에 구성합니다.
2. 비공개 테스트 서버 전용 게임가드 업데이트 서버가 존재하지 않을 경우
   - 비공개 테스트 서버에 대응하는 게임가드 업데이트 서버를 구성한 후에 ini 파일과 GameGuard.des 파일을 제공합니다.
   - 게임클라이언트는 제공해드린 ini 파일에 해당하는 게임 코드로 게임가드 생성자를 호출하도록 처리합니다. (real server 에 적용된 GameGuard 와 구분해야 합니다. 절대로 real sever 에 적용된 게임클라이언트가 테스트 서버에 구성된 GameGuard 를 호출하시면 안됩니다.)

B. 게임서버(테스트서버)를 시작시키고 테스트 서버에 접속하는 게임클라이언트를 실행시켜서 접속 테스트를 합니다.

i. 정상 접속 여부는 서버에 로그를 따로 남기시거나 게임을 약 10분 정도 플레이 하신 후 GameGuard 폴더의 *.erl 파일을 잉카측에 보내어 확인합니다.

ii. CS인증 오류(각 함수의 호출 리턴값이 ERROR_SUCCESS 가 아닌값) 인 경우 로그작성법[참고#4]을 참고하여 로그를 남긴 후에 잉카와 문제를 해결합니다.

2. CS인증 2.5 기능 테스트 (옵션)

- CS인증 2.5 는 최신 버전의 게임가드가 접속을 할 경우 일정한 조건[참고#5]이 지나면 구 버전으로 접속하는 사용자의 경우 에러(12번, 14번) 가 발생합니다. 항상 최신 버전의 게임가드를 서버에서 검사를 하므로 게임가드의 업데이트와 연계하여 신종 부정 툴에 대한 신속한 대응이 가능합니다.

가. 서버 시작 후 처음 접속하는 게임가드의 버전 확인

- CS인증 2.5 는 서버가 시작할 때 기본적으로 특정 버전(2003010101)이 설정되며 게임가드는 해당 버전보다 항상 상위버전입니다. 그러므로 서버 시작 후 게임가드가 접속하면 새로운 버전에 대한 조건을 검사하여 조건을 만족할 경우 보다 낮은 버전의 게임가드로 서버에 접속할 때 에러가 발생하게 됩니다.

A. 서버 시작 후 게임클라이언트가 접속하게 되면 항상 새 버전 접속의 통계 데이터를 수집합니다.

B. DEBUG 로그가 남겨질 경우 각 버전의 통계 수집정보가 출력됩니다.

C. 서버 시작 후 버전 업데이트 조건[참고#5]에 만족할 경우 새 버전으로 업데이트 됩니다.

나. 게임가드의 업데이트로 인한 버전 확인
- 게임가드는 새로운 해킹 기술이나 부정툴등으로 취약점이 발견되었을 경우 신속하게 업데이트를 하게 됩니다. 이때 해당 취약점의 성격에 따라 CS인증 버전을 올려서 업데이트 하는 경우가 있습니다. 이 때 서버에서는 새로운 CS인증 버전으로 접속하는 사용자에 대해서 통계 데이터를 수집하게 되고 버전 업데이트 조건을 만족할 경우 해당 버전을 업데이트 하고 그 이전의 버전으로 접속시 14번 에러가 발생합니다.

A. “가” 항목을 완료한 상태(서버를 실행시키고 현재 게임가드의 버전으로 업데이트 된 상태) 로 서버를 준비합니다.
B. 잉카측에 버전 업데이트 테스트 요청을 하게 되면 잉카에서는 게임가드의 CS인증 버전을 올린 후에 업데이트를 하게 됩니다.
C. 테스트서버용 게임클라이언트를 실행시켜서 새로운 게임가드를 업데이트 받는지 확인합니다.
   - 게임가드 업데이트 창을 눈으로 확인하시거나 게임을 플레이 해보신 후에 GameGuard 폴더의 *.erl 을 잉카측에 보내어 확인할 수 있습니다.
D. 버전 업데이트 조건을 만족할 경우 새버전이 업데이트되고 해당 사항에 대해서는 GGAuthUpdateCallback() 함수를 통해서 보고가 되며 DEBUG 로그를 남길 경우 다음과 같은 로그가 출력됩니다. “**New GG version accept[2006031702]**”
E. 잉카에 요청하여 CS인증 구버전으로 접속하여 접속시 에러가 발생하는지 확인합니다.
   - 구버전의 게임가드로 접속하는 방법은 공개할 수 없으므로 반드시 잉카에서 확인 해야함을 양해 부탁 드립니다.
   - 구버전으로 접속시 서버에서는 14번 오류가 발생하게 됩니다.

다. 게임가드의 업데이트로 인한 모듈 번호 확인
- CS인증은 버전 확인 뿐만 아니라 CS인증 모듈(Protocol)의 번호도 함께 검사합니다. 처음 인증을 시작할 때 최우선적으로 검사하는 항목은 CS인증 모듈의 번호입니다. CS인증 모듈의 번호가 잘못되었을 경우 12번 에러가 발생합니다.

A. “가” 항목을 완료한 상태(서버를 실행시키고 현재 게임가드의 버전을 허가한 상태) 로 서버를 준비합니다.
B. 잉카측에 인증 모듈 번호 업데이트 테스트 요청을 하게 되면 잉카에서는 게임가드의 CS인증 모듈 번호를 올린 후에 업데이트를 하게 됩니다.
C. 테스트서버용 게임클라이언트를 실행시켜서 새로운 게임가드를 업데이트 받는지 확인합니다.
   - 게임가드 업데이트 창을 눈으로 확인하시거나 게임을 플레이 해보신 후에 GameGuard 폴더의 *.erl 을 잉카측에 보내어 확인할 수 있습니다.
D. 버전 업데이트 조건을 만족할 경우 새버전이 업데이트되고 해당 사항에 대해서는 GGAuthUpdateCallback() 함수를 통해서 보고가 되며 DEBUG 로그를 남길 경우 다음과 같은 로그가 출력됩니다. “**NEW PROTOCOL ACTIVE!!!!!!!**”
E. 잉카에 요청하여 CS인증 구버전으로 접속하여 접속시 에러가 발생하는지 확인합니다.
   - 구버전의 게임가드로 접속하는 방법은 공개할 수 없으므로 반드시 잉카에서 확인 해야함을 양해 부탁 드립니다.
   - 구버전으로 접속시 서버에서는 12번 오류가 발생하게 됩니다.

6장 참고

[참고#1] 함수 프로토타입

- **UINT32 InitGameguardAuth(char* sGGPath, DWORD dwNumActive, BOOL useTimer, int useLog)**
  - 게임가드 CS 인증 모듈을 초기화합니다. .

  @param sGGPath : ggauth##.dll 파일의 경로 (절대경로 or 상대경로로 파일이름은 제외)
  @param dwNumActive : CS인증의 버전/모듈 업데이트 조건중의 하나인 새버전 접속자수를 의미합니다. 30분의 시간이 지나고 이 값을 넘는 유저가 게임서버에 접속을 할 경우 서버는 새로운 버전/모듈 을 허가하게 됩니다. 기능 테스트시에는 약 10 정도로 설정하고 실제 서버에 적용할 때에는 예상 동시접속자수의 30% 정도로 입력하시는 것이 좋습니다.
  @param useUpdateTimer : 게임서버에서 직접 알고리즘 변경 타이머 함수를 호출 할 경우 true 로 전달합니다. CS인증 내부에서 타이머를 생성하여 알고리즘 변경을 인증모듈 내부에서 할 경우에는 FALSE 로 전달합니다. true 로 설정을 하면 타이머가 생성하지 않으며 게임서버에서 GGAuthUpdateTimer() 를 호출해야만 알고리즘이 변경됩니다.
  @param useLog : CS인증 라이브러리 내부에서 생성하고 리포팅(NpLog() 호출)할 로그의 종류를 플래그를 통해서 설정합니다. 모든 로그 생성 : 0x3, 디버그로그 : 0x1, 에러로그 : 0x2, 로그생성안함 : 0x0
  @return : 리턴값이 0 이면 성공입니다. 나머지는 에러는 [참고#3] 를 참조하십시오.

- **void CleanupGameguardAuth()**
  - 로딩된 CS인증 모듈을 해제하고 자체 생성한 타이머를 종료시킵니다.

- **UINT32 AddAuthProtocol(char* sDllName)**
  - 게임가드 인증 모듈을 실시간으로 추가합니다. 추가할 파일은 기존에 로딩한 인증모듈과 같은 폴더에 존재해야 하며 csauth2.cfg 와 이미 추가한 모듈은 다시 추가할 수 없습니다.

  @param sDllName : 인증 모듈의 DLL 파일의 이름. (경로없이 파일 이름만 사용)
  @return : 리턴값은 0 이면 성공입니다. 나머지는 에러는 [참고#3] 를 참조하십시오.

- **UINT32 CheckCSAuth(bool bCheck)**
  - 인증 기능을 실시간으로 On/Off 할 수 있습니다.

  @param bCheck: bCheck 가 true 일 때, 인증 On, false 일 때 인증 Off.
  @return: 리턴값이 0 이면 성공입니다.

- **int ModuleInfo(char* buf, int len)**
  - 현재 서버에서 로딩한 모든 서버인증 모듈에 대한 정보를 buf 에 저장합니다.

  @param buf : 모듈 정보를 저장할 문자열 버퍼 포인트.
  @param len : buf 의 길이
  @return : 리턴값이 NPGG_INFO_SUCCESS = 0 이면 성공 입니다.

define	value	
NPGG_INFO_SUCCESS	0	성공
NPGG_INFO_ERROR_NOTENOUGHFMEMORY	1	buf 의 사이즈 부족

- **UINT32 GGAuthUpdateTimer()**
  - 서버에서 직접 알고리즘 변경 타이머 함수를 호출합니다. 이 함수를 호출할 때마다 서버에서는 알고리즘 변경여부를 검사하여 적절한 시간에 변경을 하게 됩니다. InitGameguardAuth() 함수의 3번째 전달인자가 true 로 설정되어 있어야 합니다.

  @param : 리턴값이 ERROR_SUCCESS = 0 이면 성공입니다.
  -
- **void CCSAuth2::Init()**
  - CCSAuth2 객체(csa)의 값을 초기화 시킵니다. csa 객체를 다시 사용할 경우 꼭 사용하기 전에 Init() 를 호출해야 하며 csa 를 사용자가 접속할 때 새로 생성한다면 Init() 을 호출 할 필요가 없습니다.

- **void CCSAuth2::Close()**

- CCSAuth2 객체(csa) 를 폐기합니다. 접속자가 서버의 접속을 종료하였을 경우 호출합니다. csa 객체를 메모리에서 해제시킨다면 Close() 를 호출할 필요가 없습니다.

- **UINT32 CCSAuth2::GetAuthQuery()**
  - CS인증 Query 를 생성하여 CCSAuth2.m_AuthQuery에 저장합니다.
  - 리턴값이 0 이면 성공입니다. 나머지 에러는 [참고#3]을 참고하십시오.

- **UINT32 CCSAuth2::CheckAuthAnswer ()**
  - 사용자에게 받은 인증 응답값(CCSAuth2.m_AuthQuery 에 저장한 후)을 이 함수를 호출하여 검증 합니다.
  - 리턴값이 0 이면 성공입니다. 나머지 에러는 [참고#3]을 참고하십시오.

- **UINT32 CCSAuth2::GetCSAuthState(PGG_CSAUTH_STATE m_CSAuthState)**
  - 캐릭터 서버 이동 시 현재 서버에서의 객체 정보를 저장할 때 사용합니다.
  - GG_CSAUTH_STATE 구조체를 생성하신 후 이 함수의 인자로 사용하시면 됩니다.
  - 이 함수 호출 후 저장된 정보를 이동할 서버로 보내고, 이동할 서버에서는 SetCSAuthState 함수로 정보를 저장하면 됩니다. (아래의 SetCSAuthState() 참조)

  @param m_CSAuthState: GG_AUTH_STATE 구조체로 현재 서버에서의 객체 정보를 저장
  @return: 리턴값이 0 이면 성공입니다.

- **UINT32 CCSAuth2::SetCSAuthState(PGG_CSAUTH_STATE m_CSAuthState)**
  - 캐릭터 서버 이동 시 이전 서버에서 받은 객체 정보를 현재 서버에 저장할 때 사용합니다.
  - GG_AUTH_STATE 구조체에 이전 서버로부터 받은 데이터를 저장하고 이 함수의 인자로 사용합니다.

  @param m_CSAuthState: GG_AUTH_STATE 구조체로 이전 서버에서의 객체 정보.
  @return: 리턴값이 0 이면 성공입니다.

- **UINT32 CCSAuth2::CheckUserCSAuth(bool bCheck)**
  - CCSAuth2 객체(csa)의 인증기능을 실시간으로 On/Off 할 수 있습니다.

  @param bCheck: bCheck 가 true 일 때, 인증 On, false 일 때 인증 Off.
  @return: 리턴값이 0 이면 성공입니다.

- **int CCSAuth2::Info(char* buf, int len)**
  - 접속중인 사용자가 사용하는 서버인증 모듈의 정보를 buf 에 저장 합니다.

  @param buf : 모듈 정보를 저장할 문자열 버퍼 포인트.
  @param len : buf 의 길이
  @return : 리턴값이 NPGG_INFO_SUCCESS = 0 이면 성공 입니다.

define	value	
NPGG_INFO_SUCCESS	0	성공
NPGG_INFO_ERROR_NOTENOUGHFMEMORY	1	buf 의 사이즈 부족

- **int CCSAuth2:: CheckUpdated ()**
  - 접속중인 사용자의 CS인증 버전 정보를 확인합니다.

  @return : 접속중인 사용자의 CS인증 버전 확인 값

define	value	
NPGG_CHECKUPDATED_VERIFIED	0	현재 서버의 버전과 같은 사용자
NPGG_CHECKUPDATED_NOTREADY	1	버전검사를 하기 전. 버전에 대한 정보가 없음.
NPGG_CHECKUPDATED_HIGH	2	현재 서버의 버전보다 상위 버전 사용자
NPGG_CHECKUPDATED_LOW	3	현재 서버의 버전보다 하위 버전 사용자

- **void CCSAuth2::AllowOldVersion()**

  - 유저가 서버이동을 할 경우 이동하는 서버의 버전이 높은 경우 접속이 끊기도록 되어 있습니다. 서버이동을 한 유저에 한해서 이 함수를 호출 할 경우 구버전 접속에 의한 에러는 발생시키지 않습니다. 첫 인증을 하기 전에 서버이동을 할 경우 서버인증이 우회가 되는 경우도 발생할 수 있으니, 이 점에 유의 하시기 바랍니다. 이 함수의 사용은 권장하지 않습니다.

- **LPGGAUTH GGAuthCreateUser()**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. CCSAuth2 클래스의 인스턴스를 내부에서 생성하여 리턴합니다. CCSAuth2 *csa = new CCSAuth2(); 와 같습니다.

@return : csa 의 인스턴스 포인터 입니다.

- **UINT32 GGAuthDeleteUser(LPGGAUTH pGGAuth)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스의 인스턴스를 내부에서 메모리 해제 시킵니다. delete csa; 와 같습니다.

@param pGGAuth : 해제시킬 CCSAuth2 의 인스턴스.
@return : 리턴값은 0 이면 성공입니다.

- **UINT32 GGAuthInitUser (LPGGAUTH pGGAuth)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스의 인스턴스를 초기화 시킵니다. csa.Init(); 와 같습니다.

@param pGGAuth : 초기화 시킬 CCSAuth2 의 인스턴스.
@return : 리턴값이 0 이면 성공입니다.

- **UINT32 GGAuthCloseUser (LPGGAUTH pGGAuth)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스의 인스턴스에 할당된 값을 해제(종료) 시킵니다. csa.Close(); 와 같습니다.

@param pGGAuth : 인증 쿼리를 생성할 CCSAuth2 의 인스턴스.
@return : 리턴값이 0 이면 성공입니다.

- **UINT32 GGAuthGetQuery (LPGGAUTH pGGAuth, PGG_AUTH_DATA pAuthData)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스의 인스턴스에서 인증 쿼리를 생성한 후에 pAuthData 에 대입합니다. csa.GetAuthQuery(); 와 같습니다.

@param pGGAuth : 인증 쿼리를 생성시킬 CCSAuth2 의 인스턴스.
@param pAuthData : 생성한 인증 쿼리를 저장시킬 인증 패킷 구조체 포인터
@return : 리턴값이 0 이면 성공입니다. 나머지 에러는 [참조#3]을 참고하십시오.

- **UINT32 GGAuthCheckAnswer(LPGGAUTH pGGAuth, PGG_AUTH_DATA pAuthData)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스의 인스턴스에서 pAuthData 의 인증 응답을 검증합니다. csa.CheckAuthAnswer(); 와 같습니다.

@param pGGAuth : 인증 응답을 검사할 CCSAuth2 의 인스턴스.
@param pAuthData : 인증을 검사할 응답 패킷 구조체 포인터
@return : 리턴값이 0 이면 성공입니다. 나머지 에러는 [참조#3]을 참고하십시오.

- **UINT32 GGAuthGetState(LPGGAUTH pGGAuth, PGG_AUTH_STATE pAuthState)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용함수.
  - 서버 이동 시 현재 서버에서의 객체 정보를 저장할 때 사용합니다.
  - csa.GetCSAuthState(PGG_AUTH_STATE pAuthState) 와 같습니다.

@param pGGAuth : 객체 정보를 저장할 CCSAuth2 의 인스턴스
@param pAuthState : 현재 서버에서의 객체 정보를 저장
@return : 리턴값이 0 이면 성공.

- **UINT32 GGAuthSetState(LPGGAUTH pGGAuth, PGG_AUTH_STATE pAuthState)**
  - 서버 이동 시 이전 서버에서 전달된 객체 정보를 현재서버에 설정할 때 사용
  - csa.SetCSAuthState(PGG_AUTH_STATE pAuthState) 와 같습니다.

@param pGGAuth : 객체 정보를 설정할 CCSAuth2 의 인스턴스
@param pAuthState : 이전 서버에서의 객체 정보
@return : 리턴값이 0 이면 성공

- **UINT32 GGAuthCheckUserCSAuth(LPGGAUTH pGGAuth, bool bCheck)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스의 인스턴스의 인증기능을 실시간으로 On/Off 할 수 있습니다. csa.CheckUserCSAuth(bool bCheck); 와 같습니다.

@param pGGAuth : 인증 기능을 설정할 CCSAuth2 의 인스턴스.
@param bCheck : true 이면 인증 On, false 이면 인증 Off
@return : 리턴값이 0 이면 성공입니다. 나머지 에러는 [참조#3]을 참고하십시오.

- **int GGAuthCheckUpdated (LPGGAUTH pGGAuth)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스 인스턴스가 현재 게임서버에 적용된 CS인증 버전/모듈과 비교합니다. csa.CheckUpdated() 와 같습니다.

  @param pGGAuth : 인증 버전을 확인할 CCSAuth2 의 인스턴스.
  @return : 리턴값이 0 이면 현재 서버의 버전을 사용중 입니다.

define	value	
NPGG_CHECKUPDATED_VERIFIED	0	현재 서버의 버전과 같은 사용자
NPGG_CHECKUPDATED_NOTREADY	1	버전검사를 하기 전. 버전에 대한 정보가 없음.
NPGG_CHECKUPDATED_HIGH	2	현재 서버의 버전보다 상위 버전 사용자
NPGG_CHECKUPDATED_LOW	3	현재 서버의 버전보다 하위 버전 사용자

- **int GGAuthUserInfo (LPGGAUTH pGGAuth , char* dest, int length)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스 인스턴스가 사용하는 서버인증 모듈의 정보를 dest 에 저장 합니다. csa.Info() 와 같습니다.

  @param pGGAuth : 서버인증 모듈 정보를 확인할 CCSAuth2 의 인스턴스.
  @param dest : 모듈 정보를 저장할 문자열 버퍼 포인트.
  @param length : dest 의 길이
  @return : 리턴값이 NPGG_INFO_SUCCESS = 0 이면 성공 입니다.

- **UINT32 GGAuthGetUserValue (LPGGAUTH pGGAuth, int type)**
  - CCSAuth2 클래스를 사용할 수 없는 C 로만 구현된 게임서버를 위해 제공되는 C 전용 함수입니다. 넘겨받은 CCSAuth2 클래스 인스턴스가 갖고 있는 내부 값을 리턴합니다. csa.m_AuthQuery, csa.m_AuthAnswer 에 바로 접근할 수 없으므로 이 함수를 통해서 값을 확인할 수 있습니다.

  @param pGGAuth : 서버인증 내부 값을 확인할 CCSAuth2 의 인스턴스.

  @type : 필요한 인증값을 지정할 플래그. 인증쿼리/응답 | 값 종류 로 구성됩니다.

	NPGG_USER_AUTH_QUERY	NPGG_USER_AUTH_ANSWER
NPGG_USER_AUTH_INDEX	0x01 | 0x10 = 0x11	0x02 | 0x10 = 0x12
NPGG_USER_AUTH_VALUE1	0x01 | 0x20 = 0x21	0x02 | 0x20 = 0x22
NPGG_USER_AUTH_VALUE2	0x01 | 0x40 = 0x41	0x02 | 0x40 = 0x42
NPGG_USER_AUTH_VALUE3	0x01 | 0x80 = 0x81	0x02 | 0x80 = 0x82

  @return : 플래그에 해당하는 인증 값을 리턴 합니다. 에러가 발생했을 경우 -1 을 리턴합니다.

- **UINT32 SetGGVerLimit(unsigned long nLimitVer)**
  - CS인증 버전의 상한치를 제한합니다. 해당 상한치는 csver.cfg 파일에 남겨지며 서버 시작시 자동 로딩됩니다.

  @param nLimitVer: 제한할 상한치 버전. 허용범위 2003010101 ~ 2100010101
  @return : 리턴값이 NPGG_INFO_SUCCESS = 0 이면 성공 입니다.

- **UINT32 SetUpdateCondition(int nTimeLimit, int nCondition)**
  - CS인증 버전 업데이트 조건을 설정합니다.

  @param nTimeLimit: 업데이트 조건 중 통계데이터를 수집하는 시간입니다. 해당시간 동안 수집된 데이터의 통계로 업데이트를 결정합니다. 기본값은 30(분) 이며 0 보다 큰 숫자만 입력 가능합니다. 주의) 너무 낮은 수치로 입력할 경우 버전업데이트 시간간격이 너무 좁아서 정상사용자가 피해를 입을 수 있습니다.)
  @param nCondition: 업데이트 조건 중 통계데이터 중에 선택할 조건을 지정합니다. 여러 버전의 통계데이터중에 nCondition (%) 이상인 버전을 업데이트 하게 됩니다. 기본값은 50(%) 이며 1 ~ 100 까지의 숫자를 입력 가능하지만 50% ~ 70% 정도로 설정하는 것이 안전합니다 .
  @return : 리턴값이 NPGG_INFO_SUCCESS = 0 이면 성공 입니다.

- **int DecryptHackData(char* lpszUserKey, LPVOID lpData, DWORD dwLength)**
  - 게임가드의 LPBYTE GetHackInfoFromGameMon(DWORD* dwSize) 함수에서 return 된 암호화된 Hack Log 를 복호화 합니다.

  @param lpszUserKey: 암호화된 고유키를 입력합니다. NPGameLib.lib 의 CNPGameLib 생성자에서 입력받은 LPCSTR lpszGameName 을 입력하시면 됩니다.
  @param lpData: GetHackInfoFromGameMon 함수에서 return 된 data 를 입력합니다.
  @param dwLength: GetHackInfoFromGameMon 함수의 dwSize 값을 입력합니다.
  @return : 복호화된 data 의 size 를 return 합니다. -1 일경우 실패를 의미합니다.

[참고#2] CS인증 주기

- CS인증은 서버에서 게임가드의 정상 동작여부를 확인하는 것이 최종 목적입니다. 하지만 게임가드가 동작하지 않는 상황과 CS인증의 주기를 교묘히 이용하는 부정 툴이 발견되었습니다. 다음은 CS인증의 취약점 중에 하나인 인증 주기를 보안하도록 설정하는 방법입니다. CS인증의 주기는 게임서버에서 어떻게 처리하느냐에 따라 결정됩니다.
- CS인증 패킷은 첫번째 패킷과 두번째 이후의 패킷으로 구분됩니다. 첫번째 패킷은 CS인증 버전을 확인하는 절차로 서버에서 동일한 패킷을 보내며 게임가드는 항상 동일한 패킷(게임가드 버전을 포함)을 응답합니다. 그러므로 패킷만을 분석한다면 쉽게 해커에 의해서 첫번째 패킷은 분석될 수 있습니다. 두번째 이후의 패킷은 서버에서 키값을 항상 다르게 구성해서 클라이언트에게 보냅니다. 그러므로 두번째 이후의 인증 패킷이 실제 인증 패킷이며 정확한 인증을 위해서는 두번째 이후의 인증이 정확해야만 합니다.
- CS인증 패킷이 첫번째 와 두번째 이후의 패킷이 구분됨으로써 해커는 첫번째 인증패킷과 두번째 인증 패킷의 주기 사이를 노리는 방법을 사용합니다. 부정 툴은 게임가드를 동작시키지 않고 두번째 인증 패킷을 검사하여 서버에서 접속을 해제하기 전까지 사용이 가능합니다. 부정툴은 끊임없이 서버에 재접속해서 두번째 인증 패킷을 확인하는 시간까지 사용함으로써 결과적으로 정상적인 사용이 가능해 집니다.
- CS인증 주기를 다음과 같이 처리하시기를 권장합니다. **서버에서 CS인증을 2회 이상 인증 받지 않으면 다음으로 진행하지 않도록 처리하십시오.** 서버에서는 사용자의 접속(로그인)이 끝난후 바로 CS인증 첫번째 인증 패킷을 전송합니다. 10초 이내에 응답을 하지 않으면 접속을 해제시키며 응답이 도착할 경우 두번째 인증 패킷을 **즉시** 전송합니다. 그 이후에는 타이머를 생성해서 주기적으로(3분~5분) 인증 패킷을 생성해서 전송합니다.

[참고#3] 에러코드 정리

성공	0	함수 호출 결과 성공
초기화 실패	1	메모리 할당 실패
	2	ggauth##.dll 로드 실패
	3	ggauth##.dll의 Export 함수 가져오기 실패
	4	ggauth##.dll이 초기화되기 전에 Export 함수 호출하였음
	5	csauth2.cfg 파일로부터 설정 로딩 실패.
인증 실패	10	함수 호출 시 invalid parameter 전달
	11	인증 Query에 대한 클라이언트의 응답 없음
	12	클라이언트의 인증 프로토콜 버전 틀림
	13	인증 Query에 대한 클라이언트의 응답값이 틀림
	14	게임가드 버전 검사 실패. 최신버전의 게임가드를 사용하지 않음. * csa 객체를 처음사용할 때 GetAuthQuery() 를 연속으로 2번 호출 시 발생
Gamemon 버전 틀림	101	Gamemon 버전이 틀림
	102	Gamemon 버전 확인위한 코드값이 틀림

[참고#4] CS인증 실패 오류 로그 작성법.

- CS인증의 인증 쿼리 생성(GetAuthQuery()) 와 검증(CheckAuthAnswer()) 을 할 때에 발생하는 인증 오류에 대해 로그를 남기면 인증 실패가 발생한 원인을 파악할 수 있습니다.
- CS인증에 대한 오류는 1차로 리턴값을 통해서 확인할 수 있습니다. 좀더 정확한 분석을 위해서는 인증을 할 때 보낸값과 받은값을 **해당사용자의 접속정보**와 함께 남기는 것입니다. CS인증의 주고 받은 값은 csa 객체 내부에서 저장하고 있으므로 이 값을 출력 하십시오.

  * CCSAuth2 를 사용할 수 있는 환경 : m_AuthQuery 는 인증쿼리, m_AuthAnswer 는 인증응답값이며 _GG_AUTH_DATA 구조체 DWORD 값 4개 로 구성되어 있습니다.

  ex) sprintf(buf, " [%02d:%02d:%02d] [ERRCODE:%d] Query : %08X %08X %08X %08X [UserID : %s] ", NowTime.hour, NowTime.min, NowTime.sec, dwReturn, csa.m_AuthQuery.dwIndex, csa.m_AuthQuery.dwValue1, csa.m_AuthQuery.dwValue2, csa.m_AuthQuery.dwValue3, m_UserID);
  sprintf(buf, " [%02d:%02d:%02d] [ERRCODE:%d] Answer : %08X %08X %08X %08X [UserID : %s] ", NowTime.hour, NowTime.min, NowTime.sec, dwReturn, csa.m_AuthAnswer.dwIndex, csa.m_AuthAnswer.dwValue1, csa.m_AuthAnswer.dwValue2, csa.m_AuthAnswer.dwValue3, m_UserID);

  * CCSAuth2 를 사용할 수 없는 환경 : GGAuthGetUserValue() 함수를 이용해서 CS인증 값을 가져올 수 있습니다.

  ex) sprintf(buf, " [%02d:%02d:%02d] [ERRCODE:%d] Query : %08X %08X %08X %08X [UserID : %s] ", NowTime.hour, NowTime.min, NowTime.sec, dwReturn, GGAuthGetUserValue(csa, 0x11), GGAuthGetUserValue(csa, 0x21), GGAuthGetUserValue(csa, 0x41), GGAuthGetUserValue(csa, 0x81), m_UserID);
  sprintf(buf, " [%02d:%02d:%02d] [ERRCODE:%d] Answer : %08X %08X %08X %08X [UserID : %s] ", NowTime.hour, NowTime.min, NowTime.sec, dwReturn, GGAuthGetUserValue(csa, 0x12), GGAuthGetUserValue(csa, 0x22), GGAuthGetUserValue(csa, 0x42), GGAuthGetUserValue(csa, 0x82), m_UserID);

  [Type1] - Query 값과 Answer 값을 같이 남기는 경우

  " [20:20:20][ERRCODE:0] Query : 29DD954E 77C39CFC 97ADB620 07BDE0F7, Answer: 53EA713C DAFB09A9 A6DB884E 3C8C290D, UserIdx: 1043448 "

[Type2] - Query 값과 Answer 값을 따로 남기는 경우
“ [20:20:20][ERRCODE:XX] Query : 29DD954E 77C39CFC 97ADB620 07BDE0F7 [UserID : XXXX] “
“ [20:20:20][ERRCODE:XX] Answer : 53EA713C DAFB09A9 A6DB884E 3C8C290D [UserID : XXXX] “

[참고#5] CS인증 버전 업데이트

- 게임가드는 신종 부정툴에 대해서 실시간 업데이트를 통해 신속히 대응 하고 있으며 언제나 최신 게임가드는 보고된 모든 부정툴에 대해 클라이언트 보안을 책임지고 있습니다. 하지만 게임가드의 버전을 게임클라이언트와 게임가드와의 검사만으로 행해지던 과거의 취약점을 이용하여 수많은 구버전을 이용한 bypass 방법이 발견되었습니다. 따라서 게임서버에서 게임가드의 버전을 검사함으로써 원천적으로 구버전 게임가드를 이용한 bypass 를 차단할 수 있습니다.
- CS인증 2.5 는 항상 최신 버전의 게임가드를 유지합니다. 게임서버에서는 CS인증 패킷중에 첫번째 패킷을 통해서 게임가드의 버전을 알아내며 현재 서버에 설정된 이전 버전의 접속자는 CheckAuthAnswer() 에서 14번(CS인증 버전) 혹은 12번(CS인증 프로토콜) 오류를 리턴하게 됩니다.
- 만일 게임가드는 실시간으로 업데이트 되었을 경우 게임서버는 게임가드의 버전 데이터를 통계 데이터로 분류하여 자동으로 새버전을 받아들이며 구버전의 접속을 제한하게 됩니다.
- 게임가드의 버전과 관련된 데이터는 다음과 같습니다.

1. InitGameguardAuth() 함수의 2번째 인자인 dwNumActive
2. SetGGVerLimit() 함수의 1번째 인자인 nLimitVer
3. SetUpdateCondition() 함수의 1번째 인자 nTimeLimit 와 2번째 인자 nCondition

- 버전 업데이트 과정은 다음과 같습니다.

1. 게임가드의 업데이트(CS인증 버전 or Protocol 업데이트) 를 합니다.
2. 서버에서 처음으로 새로운 버전 접속 감지하고 통계데이터 수집 시작합니다.
3. 접속한 사용자의 버전은 통계 데이터에 쌓이게 됩니다.
4. nTimeLimit (분) 뒤에 통계 데이터를 검사하여 가장 접속자가 많은 버전이 전체 버전 목록의 nCondition(%) 를 넘을 경우 해당 버전을 업데이트 합니다. 단 이때 해당데이터는 최소 dwNumActive 번 이상 접속을 해야만 합니다.
5. 4단계 이후부터 새로 설정된 이전 버전은 버전 검사 실패를 하게 됩니다.

- DEBUG 로그에 해당 버전의 카운팅 정보가 표시됩니다.

```
[C$][S lib] UpdateGGVersion() : GGVer < new Version[2006060802] NEW VERSION count up [1]:[2006060802][2][116…
[C$][S lib] UpdateGGVersion() -----GG Ver Update RESULT-----
[C$][S lib] [0]:[2005110802][8][1167913045][1167913025],
[C$][S lib] [1]:[2006060802][2][1167913111][1167913061],
[C$][S lib] CURRENT VER : 2003010101
[C$][S lib] UpdateGGVersion() ------------------------------
[C$][S lib] UpdateProtocol() : PrtcVer < new Version[10056] NEW VERSION count up [0]:[10056][2][116791…
[C$][S lib] UpdateProtocol() -----Protocol Ver Update RESULT-----
[C$][S lib] [0]:[10056][2][1167913111][1167913061],
[C$][S lib] CURRENT VER : 10050
[C$][S lib] UpdateProtocol() ------------------------------------
```

- ERROR 로그에 버전이 업데이트 되는 정보가 표시됩니다.

```
[C$][S lib] UpdateGGVersion() : New GGVer[2005110802] find that it accepted condition[9/11]  [81%] > [70%]
[C$][S lib] UpdateGGVersion() -----GG Ver Update RESULT-----
[C$][S lib] [0]:[2005110802][9][1167913353][1167913025]: 81%,
[C$][S lib] [1]:[2006060802][2][1167913111][1167913061]: 18%,
[C$][S lib] [2]:[0][0][0][0]: 0%,
[C$][S lib] [3]:[0][0][0][0]: 0%,
[C$][S lib] [4]:[0][0][0][0]: 0%,
[C$][S lib] CURRENT VER : 2003010101
[C$][S lib] UpdateGGVersion() ------------------------------
[C$][S lib] UpdateGGVersion() : New GG version accept[2005110802] !!!!!!!!!!!!!!!!!
[C$][S lib] UpdateProtocol() : New PrtcNew[10056] find that it accepted condition[6/6] and [100%] > [70%]
[C$][S lib] UpdateProtocol() -----GG Ver Update RESULT-----
[C$][S lib] [0]:[10056][6][1167913825][1167913061]: 100%,
[C$][S lib] [1]:[0][0][0][0]: 0%,
[C$][S lib] [2]:[0][0][0][0]: 0%,
[C$][S lib] [3]:[0][0][0][0]: 0%,
[C$][S lib] [4]:[0][0][0][0]: 0%,
[C$][S lib] CURRENT PRTC : 10050
[C$][S lib] UpdateProtocol() ------------------------------
[C$][S lib] UpdateProtocol() : NEW PROTOCOL ACTIVE!!!!!!! [0x10056]:[0][1167900179],
```

- 버전이 업데이트되면 GGAuthUpdateCallback 함수에 업데이트 정보를 담아서 호출합니다.

■ (주)잉카인터넷 게임보안 사업본부

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