2009 졸업연구 결과보고서

USB를 이용한 PC보안

USB Smart Key

팀 명 SHILED

팀 원 김 종 찬
유 주 형
김 재 식
방 세 일

2009. 10

중부대학교 정보보호학과

요 약 문

1. 연구제목

USB를 이용한 PC보안
USB Smart Key

2. 연구 목적 및 필요성

자신의 컴퓨터나 회사의 컴퓨터는 각각 중요한 정보를 다루거나 가지고 있게 됩니다. 컴퓨터가 작업을 하고 있는 상태에서 혹은 자신의 컴퓨터의 화면이 타인에게 보여지고 싶지 않을때가 있습니다. 저희의 프로젝트는 컴퓨터의 주인이 자리를 비우는 동안 타인이 자신의 컴퓨터를 마음대로 이용하지 못하도록 하는것에 주 목적이 있습니다.

3. 연구 내용

요즘 시대의 컴퓨터는 USB라는 저장 매체 혹은 장치에 관하여 표준을 적용하고 광범위하게 사용하고 있습니다. 많은 사용자들이 쓰고 있는 USB라는 저장매체를 이용해서 쉽고 간편하게 자신의 작업환경을 보호하는 것이 주 내용입니다. 그것에 암호화 기술도 적용하여 단순히 USB를 꽂고 빼는것에 만족하지 않고 더욱 보안을 강화하는 것에 초점을 맞추었습니다.

4. 연구 결과

휴대가 간편한 USB 하나로 간편하고 확실하게 모니터 보안 기능을 제공합니다.

비밀번호 설정등의 기능등을 이용해 타인이 자신의 작업에 어떠한 수정이나 변형등을 열쇠형 USB를 통해 막는 기능을 제공합니다.

목 차

표 목차

그림 목차

1. 서론

1.1 USB

USB(범용 직렬 버스, Universal Serial Bus)는 컴퓨터와 주변 기기를 연결하는 데 쓰이는 입출력 표준 가운데 하나이다. 대표적인 버전으로는 USB 1.0, 1.1, 2.0 등이 있다.

USB는 다양한 기존의 직렬, 병렬 방식의 연결을 대체하기 위하여 만들어졌습니다. 키보드, 마우스, 게임패드, 조이스틱, 스캐너, 디지털 카메라, 프린터, PDA, 저장장치 와 같은 다양한 기기를 연결하는 데 사용되고 있다. 이러한 기기 연결의 대부분은 표준 연결 방식을 이용하여 이루어지고 있다.

USB는 PC를 위하여 개발되었지만 지금은 PDA나 게임콘솔 등에서도 채택되어 사용되고 있고, USB의 전원 공급 기능을 이용하여 충전 용도로도 많이 사용되고 있다. 2009년 전세계적으로 약 20억 개의 USB 장치가 있다.

1.2 USB 시장성

국내 USB 메모리 시장은 최근 USB 메모리 수요가 폭발적으로 늘면서 지난해 200만개가 팔렸고, 시장 규모도 600억원으로 급성장했다. 업계에서는 올해 국내 시장규모가 30~40% 성장, 최대 1000억원대에 이를 것으로 추산하고 있다.

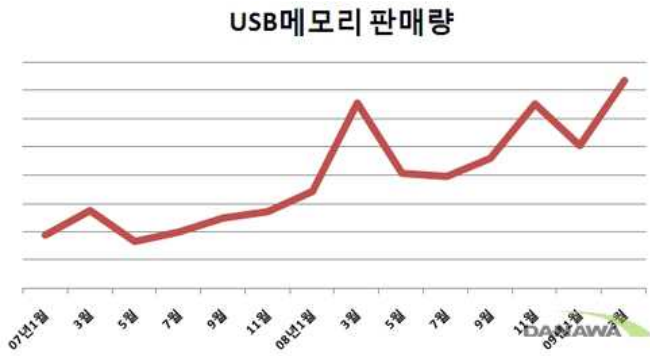


표 1-1(출처: danawa.co.kr)

이렇게 요즘 시대에 컴퓨터를 사용하는 사람이라면 누구나 한 개쯤 가지고 있는것이 바로 USB이다. 그래서 이번 프로젝트는 이렇게 광범위하게 사용되는 UBS 를 이용한 화면보안을 구상하게 되었다.

1.3 보안프로그램 현황

많은 사람들이 오고 가는 사무실에서 일하는 사람들이라면 늘 신경 쓰이는 것이 PC의 보안이다. 자리를 비운 사이 누군가 PC 모니터 화면을 볼 수도 있고, 허락도 없이 이것저것 만지고 뒤적이는 일도 생길 수 있다. 거창하게 보안이라는 말을 사용하지 않더라도 PC의 모니터 화면은 개인적인 프라이버시의 영역이기도 하다.

윈도우의 화면보호기에 비밀번호를 지정해 두면 이럴 때 유용하다. 일정 시간이 지나면 자동으로 화면보호기능이 동작하고, 다시 PC를 사용하려면 비밀번호를 입력해야 사용할 수 있기 때문이다. 물론 비슷한 기능을 제공하는 유틸리티를 사용하는 것도 방법이다.

하지만 좀 더 편리하고 완벽한 PC 보안 기능을 필요로 한다면 이 정도로는 부족하다. 그래서 민감하고 중요한 자료를 다루어야 하는 곳에서는 스마트카드나 지문 인식기 등을 이용하기도 한다.

경기 침체에 따른 전반적인 국내 기업들의 IT 투자 위축에도 불구하고, 보안 사고 대응과 규제 준수를 위한 보안 투자는 꾸준히 확대되고 있는 것으로 나타났다.

한국IDC가 최근 발간한 '국내 보안 소프트웨어 시장 전망 보고서 2009-2013'에 따르면, 2008년 국내 보안 소프트웨어 시장은 2,107억 원 규모로 집계되었으며, 올해 시장은 이보다 5.8% 증가한 2,230억 원 규모로 예상됐다. 각 기능 시장별 전망을 보면, 콘텐츠 보안 및 위협 관리(SCTM, Secure Content & Threat Management) 부문이 1,288억원, 보안 관리 및 취약점 관리(SVM, Security & Vulnerability Management) 부문이 354억 원, 사용자 계정 및 접근 권한 관리(IAM, Identity & Access Management) 부문이 364억 원, 기타 부문이 224억 원 규모에 이를 것으로 보인다.

콘텐츠 보안 및 위협 관리(SCTM) 부문의 경우, 외산 제품들의 시장 진입과 무료 온라인 백신 서비스 등 전통적인 콘텐츠 보안 소프트웨어 사업자들의 수익 구조가 악화되었으나, 작년 한해 시장의 새로운 성장 동인으로 기대되는 이머징 솔루션들의 출현이 두드러졌다. 잇단 정보 유출 사고로 DLP(Data Loss Prevention), NAC(Network Access Control), 문서보안 기술에 대한 시장의 관심이 확대되고 있으며, PC 보안의 통합 보안 추세 또한 최근 시장의 흐름을 반영하며 시장 기회를 제공하고 있다.

한편, 국내 보안 관제 서비스 사업자와 로컬 벤더를 중심으로 ESM(Enterprise Security Management), TMS(Threat Management System), PMS(Patch Management System) 등이 주도해온 국내 보안 관리 및 취약점 관리(SVM) 소프트웨어 시장은 최근 RMS(Risk Management System), SIEM(Security Information

& Event Management), 사이버 포렌식, 컴플라이언스 관리 등 정보 기반의 보다 포괄적인 보안 관리 기술로 시장의 관심이 확대되고 있으며, 나아가 RFID, CCTV, 생체인식기기 등 물리보안과 정보 보안 관제 기술이 결합한 산업보안 및 융합보안으로 시장 기회가 확대될 것으로 예상된다.

통합 보안 관리 및 서비스 수요의 증가 추세와 함께 보안 관제 서비스 제공을 위한 계정 관리(IdM) 기술이 본격 도입되기 시작하면서, 그 동안 접근 권한 관리(SSO/EAM) 중심의 국내 기업들의 IAM 인프라도 보다 포괄적인 형태로 진화할 것으로 보인다. 비용 절감 및 효율성 확보를 위한 다양한 통합 사업은 물론 컴플라이언스 솔루션 도입이 가시화되고 있는 국내 비즈니스 환경의 변화와 함께 IAM 시장은 점진적인 성장세를 이어갈 것으로 기대되고 있다.

IDC는 보고서를 통해, “장기적인 관점에서, 인터넷과 유무선 네트워크를 기반으로 IT 환경이 진화하고 다양한 보안 위협에 노출되는 현 상황이 심화되면서, 보안 지출은 꾸준히 증가할 것”으로 전제하고, “기업 보안 인프라의 고도화와 함께, SVM, IAM 부문에 대한 투자가 빠르게 확대될 것”으로 내다봤다. 국내 보안 소프트웨어 시장은 향후 5년간 연평균 10.5%의 견조한 성장세를 이어갈 것으로 예상되며, 2013년에 이르러 3,479억 원 시장을 형성할 것으로 IDC는 전망했다.

[그림] 국내 보안 소프트웨어 시장 전망, 2009 – 2013 (매출액:백만원/ 성장률:%)

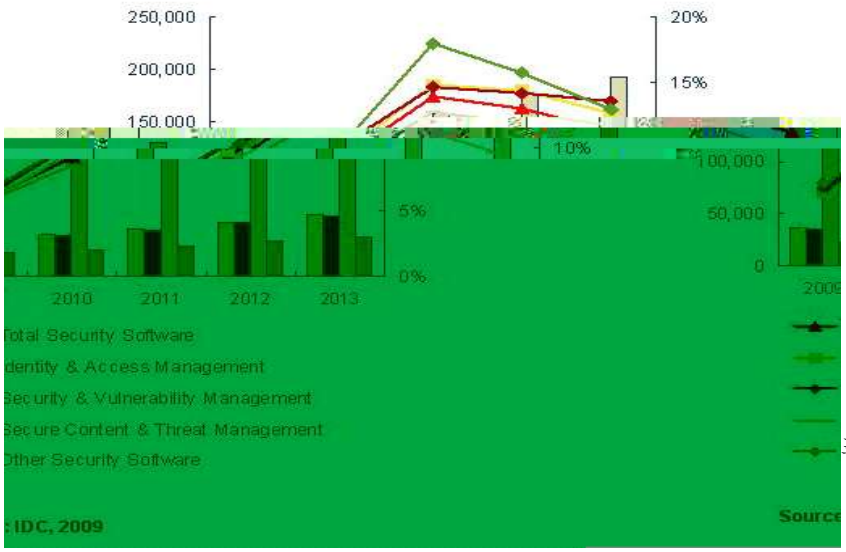


표 1-2(출처:IDC, 2009)

1.4 개인정보 유출 사례 원인

국내 개인정보 유출 사례 원인을 분석하면 내부정보 접근자에 의한 해킹이 많은 것으로 분석됐다. 따라서 이러한 USB 화면보안 프로그램이 가까운 내부정보 접근자에 의한 해킹도 차단해 줄것으로 기대된다.

표 1-3산업기밀 정보 유출자(출처 : 한국산업기술진흥원)

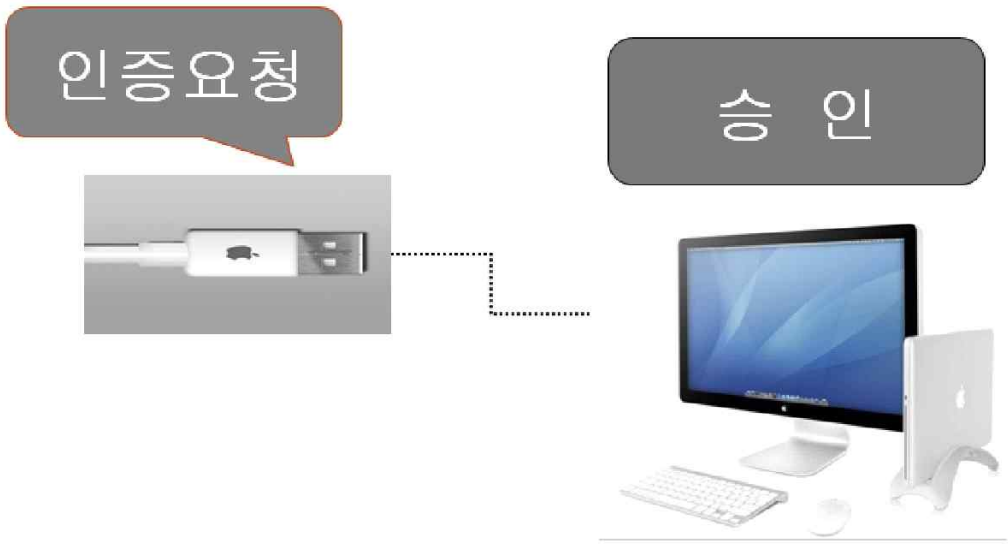
최근까지 보안의 경향은 외부로부터의 침입에 대비한 물리적인 보안 중심이었다. 방화벽, IDS, IPS, VPN, 하물며 백신까지도 모두 외부로부터의 해킹이나 바이러스, 웜 등으로부터 네트워크와 시스템을 보호하기 위한 솔루션이었던 것. 하지만 이제 외부로부터의 침입을 대비한 보안시스템들은 어느 정도 갖춰졌으며 내부 보안 역시 중요하다는 인식이 확산되고 있다.

기본 인프라만 갖추면 외부 보안은 막을 수 있지만 내부 보안은 제품 한 두 개 설치로 끝나는 것이 아니다. 더욱이 외부에서 침입시 내부 보안이 제대로 되어있지 않으면 위험은 더욱 치명적이 된다.

자신의 컴퓨터나 회사의 컴퓨터는 각각 중요한 정보를 다루거나 가지고 있게 된다. 컴퓨터가 작업을 하고 있는 상태에서 혹은 자신의 컴퓨터의 화면이 타인에게 보여지고 싶지 않을때가 있다. 이번 프로젝트는 컴퓨터의 주인이 자리를 비우는 동안 타인이 자신의 컴퓨터를 마음대로 이용하지 못하도록 하는것에 주 목적이 있다. 장기적인 관점에서 이러한 프로그램의 개발과 관심이 정보보호 산업에 발전을 기여할 것으로 기대된다.

2. 프로그램 구현 및 설명

2.1 프로그램의 메커니즘



(그림 1)

USB ON시 프로그램 절차

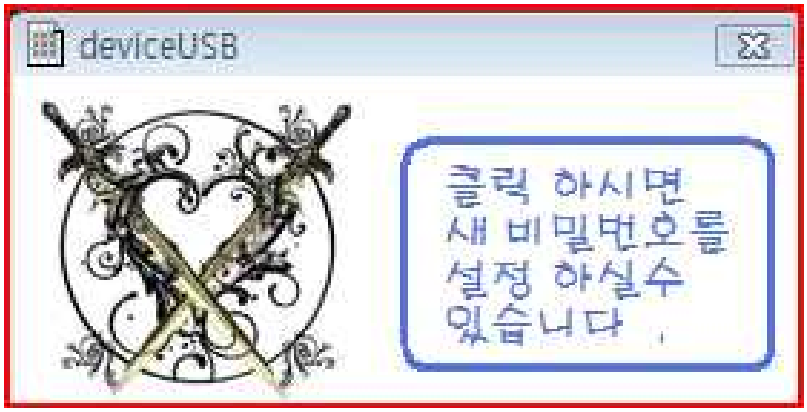
프로그램은 DBT_DEVICEARRIVAL 의 메시지를 통해 usb on 이 되었는지를 판단한후 모든 드라이브를 돌면서 해당 저장 메체를 통해 키가 있는지를 확인 한다. 만약 그 키가 없다면 해당 저장 메체는 일반 USB임을 판단하고 해당 키가 있다면 USB는 열쇠로서 인식하게 된다.

이 두 번의 인증과정을 거친후 프로그램은 키보드와 마우스의 제한을 풀게 되고 이 프로그램에서 만들었던 가짜 화의 프로그램을 진짜화면으로 다시 바꿈으로서 ON 시의 상황을 모두 마치게 된다.

USB OFF시 프로그램 절차

키가 있는 저장메체의 오프시 루틴은 우선 정상적인 키가 있음에도 불구하고 다른 키가 빠지는 상황에서 화면 제어와 키보드의 제어 권한을 뺏는 것을 방지 해야 한다. 이것은 미리 키가 들어 왔다는 전제 하에 루틴이 수행 되는데 프로그램은 KEY가 ON 시 특정 BUFFER 에 들어 왔음을 알고 OFF 시 해당 경로에서 키가 빠졋는지를 검사 하게 된다 이 인증 과정을 거친후 정상적인 키가 빠졋다면, 키보드를 제어하는 (*FuncCAD)(); 와 (*pFuncSwitching)()를 호출해 키보드의 입력을 막고 새로운 화면 보호기를 실행하게 된다. 이로서 정상적인 키가 들어올 때까지 프로그램은 어떠한 방법으로도 풀리지 않는다

2.2 프로그램의 실행화면



(그림 2)

Shield 실행화면

우리 프로그램은 총 3개의 실행 파일을 가지고 있게 된다.

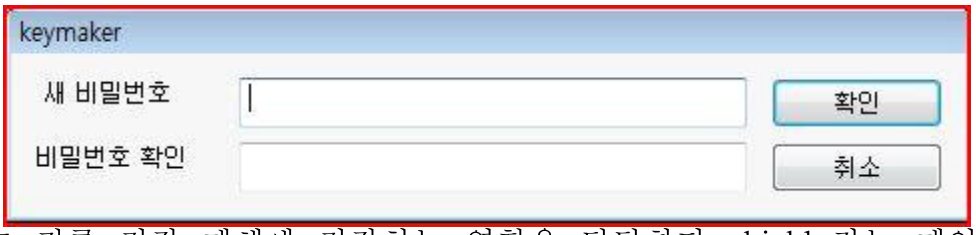
각각의 실행 프로그램들은 자신의 영역을 가지고 있어 해당 루틴을 분할하여 실행 하고 deviceusb의 프로그램은 중앙 관리 역할을 하는 프로그램 이라 볼수 있다. 이 프로그램은 usb의 키가 on 과 off 되는지 항상 판단 하고 있으며, 이 프로그램을 클릭하면 새로운 키를 만들 수 있는 keymaker 실행파일을 제어 한다.

또한 usb off 시 화면 보호 프로그램을 가동 시킨다.

Keymaker 실행화면

(그림 3)

keymaker 는 사용자가 자신만의 비밀번호를 새로 설정 하여



그 키를 저장 메체에 저장하는 역할을 담당한다 shield 라는 메인 프로그램과는 별도로 구동 되며, 암호화 및 저장을 수행한다.
루틴은 다음과 같다.

1. 4자 이상의 키만을 받아 들인다 .
2. 새롭게 입력되는 비밀번호와 새롭게 입력한 비밀 번호가 재대로 입력한 것인지 판단한다..
3. 새로운 비밀번호를 암호화 한다.
4. 키가 저장될 저장메체를 탐색한다.
5. 저장 할수 있는 저장 메체인지 검사 한후 사용자의 동의를 통하여 해당 메체에 저장한다.
6. 키 메이커를 끝낸다.

화면 잠금 실행화면

(그림 4)

화면 잠금 프로그램은 프로그램 사용자가 자리를 비울시 해당 화면을 보여줌으로써 화면에 의한 정보가 새어 나가지 않게끔 하는 것이 주 목적이다 .
이 프로그램은 별로의 실행파일로 가동 되며 해당 프로그램은 deviceusb에 의해서 제어 된다.
이 프로그램은 사용자가 임의 실행 할수 있으나, 보안적 요소가 추가되지 않은 상태이므로 쉽게 종료 될수 있다. 하지만 정상적인 루틴으로 수행시 완벽한 화면 보안을 시행한다.

3. 결론 및 향후 과제

인터넷 보안이 강조대는 시기에 많은 사용자가 이용하고 있는 USB를 통하여 개인컴퓨터의 보안 프로그램을 강화해 보았다. 그 보안의 정도에 있어서 사용자가 불편을 느끼지 않을정도와 그 정보에 대한 침해가 가능해야한다는 점에서 이번 프로그램은 그 수준을 잘 충족시켰다고 본다. 물론 완전한 프로그램이란 없듯이 더욱더 복잡한 암호와 새로 개발되는 운영체제에 이용할 수 있도록 프로그램을 계속해서 버전업 해야한다.

부록. 프로그램 소스코드 및 설명

```
//#include <stdio.h>
#include "stdafx.h"
#include "deviceUSB.h"
#include "Dbt.h"
#include "shellapi.h"
//#include "winwlx.h"
#include "Resource.h"
//#include "io.h"
//#include "time.h"
//#include    <commctrl.h>


#define MAX_LOADSTRING 100

#define USE_UNROCK 0x0000
#define _PATH_CHECK_FROM_DRIVER_PART1 ":/shield.key"
#define _PATH_CHECK_FROM_DRIVER_LOCAL "bitdata.dat"

//윈도우 관련 변수
HDC g_hMemDC;
HINSTANCE hInst;

                                        // current instance
TCHAR szTitle[MAX_LOADSTRING];
                                                    // The title bar
text
TCHAR szWindowClass[MAX_LOADSTRING];
                                        // the main window class name

// Forward declarations of functions included in this code module:
ATOM
                MyRegisterClass(HINSTANCE hInstance);
```

```
BOOL
                    InitInstance(HINSTANCE, int);
LRESULT CALLBACK                WndProc(HWND, UINT, WPARAM,
LPARAM);
INT_PTR CALLBACK                About(HWND, UINT, WPARAM,
LPARAM);

//_DLL_ int WINAPI CtrlAltDel_Enable_Disable(BOOL);



// Global Variables:
int _RESULT;
bool g_check_screen = false;
HWND g_hWnd;

//new desktop ////////////////////////////////////////////////
//new desktop 변수
HDESK           g_hOriginalThread;
HDESK           g_hOriginalInput;
HDESK           g_hNewDesktop;

STARTUPINFO                                          g_si          =
{sizeof(STARTUPINFO),};
PROCESS_INFORMATION g_pi;
//new desktop 관련 함수
void fNDT(LPWSTR szDesktopName,LPWSTR szPath);
void fClose_NDT();
BOOL StartProcess(LPWSTR szDesktopName,LPWSTR szPath);



//키 관련 상수 ////////////////////////////////////////////////
#define _MAX_CHAR 21
#define _MAX_LINE 1
//key 관련 변수
```

```
//enum {firstcheckkey = 5 , keypos1 = 10 , keypos2 = 7} g_EKeyPos;
char *key = "fkh";
char g_key_path[_MAX_CHAR];
char g_save_key_path[_MAX_CHAR];
int  g_key_pos[]={5,10,7};
bool g_key_check;
//키 관련 함수 ////////////////////////////////////////////////
bool Check_USB_key(HWND hWnd);
bool Get_Key_Path(HWND hWnd);     // 키 경로 저장
bool _Checky_key(HWND hWnd);      // 해당 경로에 키가 빠졌는지?
bool fExcuKeyMaker();

//키 메이커 관련 ////////////////////////////////////////////////////
//키 메이커 관련 변수
HANDLE g_hWnd_for_keymaker;
//키메이커 관련 함수
bool fKeyMaker();  //키가 만들어지는 과정은 이 안에서 모두 해결 되어야 한다...



//스크린 관련 ////////////////////////////////////////////////
//화면 관련 변수
HMODULE g_GetModule;
HWND g_testhwnd;
HANDLE g_hProcess = NULL;
//화면 관련 함수 ////////////////////////////////////////////////

bool LockOffScreen(UINT iMessage,WPARAM wParam);



//외부 dll 관련 함수 및 변수
//bool key_flag=false; //
```

```
HINSTANCE g_hInst_loaddll;
//int WINAPI CtrlAltDel_Enable_Disable(BOOL bEnableDisable);
int (WINAPI *pFuncCAD)(BOOL); //함수 포인터
//int WINAPI TaskSwitching_Enable_Disable(BOOL bEnableDisable);
int (WINAPI *pFuncSwitching)(BOOL); //함수 포인터
//int WINAPI Process_Desktop(char *szDesktopName, char *szPath);
int (WINAPI *pFuncProcess_Desktop)(char*,char*);//함수 포인터



////////////////////////////////////////////////////////////
//윈도우 메인..

int APIENTRY _tWinMain(HINSTANCE hInstance,
                       HINSTANCE hPrevInstance,
                       LPTSTR    lpCmdLine,
                       int       nCmdShow)
{
                       UNREFERENCED_PARAMETER(hPrevInstance);
                       UNREFERENCED_PARAMETER(lpCmdLine);

                       // TODO: Place code here.
                       MSG msg;
                       HACCEL hAccelTable;

                       // Initialize global strings
                       LoadString(hInstance, IDS_APP_TITLE, szTitle,
MAX_LOADSTRING);
                       LoadString(hInstance, IDC_DEVICEUSB, szWindowClass,
MAX_LOADSTRING);
                       MyRegisterClass(hInstance);

                       // Perform application initialization:
                       if (!InitInstance (hInstance, nCmdShow))
                       {
```

```
                    return FALSE;
                }

                hAccelTable = LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDI_ICON1));

                // Main message loop:
                while (GetMessage(&msg, NULL, 0, 0))
                {
                    if (!TranslateAccelerator(msg.hwnd,
hAccelTable, &msg))
                    {

TranslateMessage(&msg);

DispatchMessage(&msg);
                    }
                }

                return (int) msg.wParam;
}



//
//  FUNCTION: MyRegisterClass()
//
//  PURPOSE: Registers the window class.
//
//  COMMENTS:
//
//    This function and its usage are only necessary if you want this code
//    to be compatible with Win32 systems prior to the 'RegisterClassEx'
//     function that was added to Windows 95. It is important to call this
function
```

```
//      so that the application will get 'well formed' small icons associated
//      with it.
//
ATOM MyRegisterClass(HINSTANCE hInstance)
{
                        WNDCLASSEX wcex;

                        wcex.cbSize = sizeof(WNDCLASSEX);

                        wcex.style
                        = CS_HREDRAW | CS_VREDRAW;
                        wcex.lpfnWndProc    = WndProc;
                        wcex.cbClsExtra                         = 0;
                        wcex.cbWndExtra                         = 0;
                        wcex.hInstance                          = hInstance;
                        wcex.hIcon
                        =                                       LoadIcon(hInstance,
MAKEINTRESOURCE(IDI_ICON1));
                        wcex.hCursor                            =
LoadCursor(NULL, IDC_ARROW);
                        wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);


//(HBRUSH)GetStockObject(BLACK_BRUSH);
                        wcex.lpszMenuName =
NULL;//MAKEINTRESOURCE(IDC_DEVICEUSB);
                        wcex.lpszClassName = szWindowClass;
                        wcex.hIconSm                            =
LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));

                        return RegisterClassEx(&wcex);
}

//
//    FUNCTION: InitInstance(HINSTANCE, int)
```

```
//
//    PURPOSE: Saves instance handle and creates main window
//
//    COMMENTS:
//
//            In this function, we save the instance handle in a global variable
and
//          create and display the main program window.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;




    int SX,SY,_sb,_sr,y_size;

    SX= ::GetSystemMetrics(SM_CXSCREEN);
    SY= ::GetSystemMetrics(SM_CYSCREEN);
    _sb = SX+150;
    _sr = SY+150;
    y_size = 160;

    hInst = hInstance; // Store instance handle in our global variable

    hWnd                    =                    CreateWindow(szWindowClass,
szTitle,WS_CAPTION|WS_SYSMENU,// WS_OVERLAPPEDWINDOW,
        SX-280, SY-118-50, 280, 110+30, NULL, NULL, hInstance, NULL);

    //111,108
    ::ZeroMemory(g_save_key_path,sizeof(g_save_key_path));    //키가 있다면 키
경로 저장 될곳
```

```
                    g_hInst_loaddll = ::LoadLibraryA("shieldk.dll");

                    if(g_hInst_loaddll == NULL)
                    {
                                        MessageBoxW(hWnd,_T("error
0002"),_T("error"),MB_OK);
                                        return FALSE;

                    }


                    pFuncCAD                    =                    (int
(WINAPI*)(BOOL))::GetProcAddress(g_hInst_loaddll,"_CtrlAltDel_Enable_Disabl
e@4");

                    if(pFuncCAD == NULL)
                    {                   MessageBoxW(hWnd,_T("error
0003"),_T("error"),MB_OK); return FALSE; }
//////////////
                    pFuncSwitching                    =                    (int
(WINAPI*)(BOOL))::GetProcAddress(g_hInst_loaddll,"_TaskSwitching_Enable_D
isable@4");

                    if(pFuncSwitching == NULL)
                    {                   MessageBoxW(hWnd,_T("error
0004"),_T("error"),MB_OK); return FALSE; }

//////////////
//                  pFuncProcess_Desktop                    =                    (int
(WINAPI*)(char*,char*))::GetProcAddress(g_hInst_loaddll,"_Process_Desktop@8"
);

//                  if(pFuncProcess_Desktop == NULL)
//                  {                   MessageBoxW(hWnd,_T("error
0010"),_T("error"),MB_OK); return FALSE; }
```

```
//////////
                        //처음 실행을 확인 하기 위해서..
                         ZeroMemory(&g_pi, sizeof(g_pi));

                         g_hWnd = hWnd;

    if (!hWnd)
    {
        return FALSE;
    }

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}

//
//  FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)
//
//  PURPOSE:  Processes messages for the main window.
//
//  WM_COMMAND- process the application menu
//  WM_PAINT    - Paint the main window
//  WM_DESTROY  - post a quit message and return
//
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM
wParam, LPARAM lParam)
{
                        HDC MemDC;
                        HBITMAP MyBitmap,OldBitmap;
                        BITMAP              bit;
                        int bx,by;
```

```
int wmId, wmEvent;
PAINTSTRUCT ps;
HDC hdc;

static int error_code;
static int error_code_from_AT;
static int error_code_from_ND;
static bool check_key;

//SetTimer(hWnd,0,500,NULL);

PDEV_BROADCAST_HDR lpdb = (PDEV_BROADCAST_HDR)lParam;


switch (message)
{


case WM_COMMAND:

// 0.5 초마다 채크 해서 .

wmId    = LOWORD(wParam);
wmEvent = HIWORD(wParam);
// Parse the menu selections:
//SetTimer(hWnd,0,1000,NULL);
//srand((unsigned)time(NULL));
switch (wmId)
{
case IDM_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
break;
```

```
                                        case IDM_EXIT:
DestroyWindow(hWnd);
                                                    break;
                                        default:
                                                    return
DefWindowProc(hWnd, message, wParam, lParam);
                                        }
                                        break;

//                      case WM_TIMER:
                                        //MessageBoxW(hWnd,_T("0.5초마다
호출 안할꺼임? "),_T("error"),MB_OK);
                        /*
                                        if(g_hProcess == NULL) // 프로세스
가 죽었다면
                                        {
if(g_check_screen == true) //스크린샷은 내쪽 프로그램에서 정상적으로 죽였는
가?
                                                    {

                        //아직 true 라면 강제로 죽은것이므로 스크린 롹을다시건
다
                        StartLockScreen();
                                                    }

                                        }
                        */
//                                      break;
                                        //return 0;
                        case WM_PAINT:
                                        hdc = BeginPaint(hWnd, &ps);
```

```
                                        MemDC = CreateCompatibleDC(hdc);
                                        MyBitmap                    =
LoadBitmap(hInst,MAKEINTRESOURCE(IDB_MAIN));
                                        OldBitmap                   =
(HBITMAP)SelectObject(MemDC,MyBitmap);


GetObject(MyBitmap,sizeof(BITMAP),&bit);
                                        bx = bit.bmWidth;
                                        by = bit.bmHeight;


BitBlt(hdc,0,0,bx,by,MemDC,0,0,SRCCOPY);

                                        SelectObject(MemDC,OldBitmap);
                                        DeleteObject(MyBitmap);
                                        DeleteDC(MemDC);

                                        EndPaint(hWnd, &ps);
                                        break;
                    case WM_LBUTTONDOWN:

                                        fKeyMaker();

                                        //Get_Key_Path(hWnd);
                                        //Check_USB_key(hWnd);

                                        break;

                    case WM_RBUTTONDOWN:

//if(!Check_USB(hWnd)) //최종 버전에서는 메세지 무시.. 일부러..
                                                    /           /
                    {
```

```
                                                    /           /
                                    MessageBoxW(hWnd,_T("정상적인
키가 아닙니다 "),_T("error"),MB_OK);
                                                    /           /

                                                    /           /
                                    break; //다음 코드를 수행하지 않고
빠져 나간다.
                                                    /           /
                    }else
                                                    /           /
                    { MessageBoxW(hWnd,_T("정상적인 키 입니다. 화면 보
호 해제"),_T("ok"),MB_OK); }


                                    break;

                    case WM_DESTROY:
                                    //KillTimer(hWnd,0);
                                    FreeLibrary(g_hInst_loaddll);
                                    PostQuitMessage(0);
                                    break;

                    case WM_DEVICECHANGE:

                                    switch(wParam)
                                    {
                                    case DBT_DEVICEARRIVAL:

if(lpdb->dbch_devicetype == DBT_DEVTYP_VOLUME) //volum 이 있는 드라
이버 인가?
                                                    {

                    //강제로 다음 코드 수행.. 어떻게?
```

```
                    //usb on 시 수행되는 코드

                    //타 usb와 인식 방법 루트 수행.
                                                                    //새로 받아
온 열쇠인가?

                    //이컴에 맞는 열쇠인가?

                    //화면 잠금을 해제 한다.



                    //Sleep(2000); //오토런 인식 타임..

                    //bool temp;

                    //키가 있는 열쇠 인가??

                    //키를 찾는 부분은 usb가 on시에만 들오 오게 된다.

                    if(Get_Key_Path(hWnd))  //함수 호출이 true 라면..

                    {

                                        if(Check_USB_key(hWnd)) //키 값이
정상이라면?

                                        {


//SystemParametersInfo(SPI_SETSCREENSAVERRUNNING,  FALSE,  NULL,
NULL);

                                                            //Sleep(3000);
```

```
//MessageBoxW(hWnd,_T("키를 찾았습니다"),_T("잇힝"),NULL);
                                                    //정상적인 키가
꼽혓다
                                                    g_key_check
=true;
                                                    g_check_screen
= false;

strcpy(g_save_key_path,g_key_path); //정상 적인 키 값이므로

                                    //              if(g_pi.hProcess
!= NULL)
                                    //              {



                                    //              }


                                                    g_hProcess    =
g_pi.hProcess;
                                                    if(g_hProcess
!= NULL)  //프로세스 강제 종료..
```

```
                                                    {

                //      CloseHandle(g_pi.hProcess);

                CloseHandle(g_pi.hThread);

                TerminateProcess(g_hProcess,0);

                g_hProcess= NULL;



                                                    }

                                                    fClose_NDT();
//스위칭!

                                                    //key_flag      =
true; //정상적인 키가 들어 왔으므로 on
////////////////////////////////////////////////////////////////////////
```

```
                                                        if(pFuncCAD
== NULL){MessageBoxW(hWnd,_T("error 0000 "),_T("ok"),MB_OK);

                                                        break;}

                                                        error_code =
(*pFuncCAD)(FALSE);

                                                        //if(error_code
== 2){MessageBoxW(hWnd,_T("컨알텔 해제"),_T("error"),MB_OK);}

                                                        if(error_code ==
0){MessageBoxW(hWnd,_T("error 0001 "),_T("error"),MB_OK);

                                                        break;}
////////////////////////////////////////////////////////////////////////


if(pFuncSwitching == NULL){MessageBoxW(hWnd,_T("error 0004
"),_T("ok"),MB_OK);

                                                        break;}


error_code_from_AT = (*pFuncSwitching)(TRUE);  //알 텝을 푼다


if(error_code_from_AT == 0){MessageBoxW(hWnd,_T("error 0005
"),_T("error"),MB_OK);

                                                        break;}
////////////////////////////////////////////////////////////////////////

                                                        //g_hProcess =
::FindWindowA(NULL,"check.exe");
```

//윈도우를 다시 그려 깨끗하게 만든다..

InvalidateRect(NULL,NULL,FALSE);

//::UpdateWindow(NULL);

}

}

}// end of if(Check_USB_key(hWnd)) //키 값이 정상이라면?

break;

case DBT_DEVICEREMOVECOMPLETE:

if(lpdb->dbch_devicetype == DBT_DEVTYP_VOLUME)

{

if(g_key_check) //정상적인 키가 있었다면..

{

```
                                        if(_Checky_key(hWnd))  //키가 있었
던 경로에서 키를 찾을수 있는가?

                                        { //키가 없다면 true 리턴 했으므로
밑의 코드 수행..

                                                //키 변조는 검
사 하지 않는다, 해당 키 변조는 이미 수행 했으므로..

//MessageBoxW(hWnd,_T("해당 루틴에서 키를 찾지 못했으므로 LOCK
ON"),_T("잇힝"),MB_OK);


                                                //화면이 정상적
으로 꺼졌으므로

                                                g_check_screen
= true; //켜져 있다.


                        g_key_check = false;  //정상 적인 키가 빠졌다..

////////////////////////////////////////////////////////////////////////////////

                        if(pFuncCAD == NULL){MessageBoxW(hWnd,_T("error
0000 "),_T("error"),MB_OK);

                        break;}
```

```
                    error_code = (*pFuncCAD)(TRUE);

                    //if(error_code == 1){MessageBoxW(hWnd,_T("컨알렐 잠
금"),_T("error"),MB_OK);}

                    if(error_code == 0){MessageBoxW(hWnd,_T("error
0001"),_T("error"),MB_OK);

                    break;}
////////////////////////////////////////////////////////////////////////////////

                    if(pFuncSwitching ==
NULL){MessageBoxW(hWnd,_T("error 0004 "),_T("error"),MB_OK);

                    break;}

                    error_code_from_AT = (*pFuncSwitching)(FALSE);  //알
렙을 잠근다

                    if(error_code_from_AT ==
0){MessageBoxW(hWnd,_T("error 0005 "),_T("error"),MB_OK);

                    break;}
////////////////////////////////////////////////////////////////////////////////
```

```
                    //error_code_from_ND                                    =
(*pFuncProcess_Desktop)("new","check.exe");



                    //if(error_code_from_ND                                 ==
0){MessageBoxW(hWnd,_T("error 0011 "),_T("error"),MB_OK);

                    //break;}



                    //StartLockScreen();

                    //새로운 데스크탑 생성..

                    fNDT(_T("new"),_T("check.exe"));

                    //devment.scr

                    //StartProcess(_T("new"),_T("check.exe"));



                                        }//end   of   if(Get_Key_Path(hWnd))
//함수 호출이 true 라면..
```

```
                }//end of if(g_key_check)    //함수 호출이 true 라면..


                                                }
                                                break;
                                        }
                                        break;//WM_DEVICECHANGE

                default:
                                        return          DefWindowProc(hWnd,
message, wParam, lParam);
                }
                return 0;
}

// Message handler for about box.
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM
wParam, LPARAM lParam)
{
                UNREFERENCED_PARAMETER(lParam);
                switch (message)
                {
                case WM_INITDIALOG:
                                        return (INT_PTR)TRUE;



                case WM_COMMAND:
                                        if (LOWORD(wParam) == IDOK ||
LOWORD(wParam) == IDCANCEL)
                                        {

EndDialog(hDlg, LOWORD(wParam));
                                                r e t u r n
(INT_PTR)TRUE;
```

```
                                        }
                                        break;
                        }
                        return (INT_PTR)FALSE;
}


bool fExcuKeyMaker()
{
                        //관리자 권한으로 실행
                        //TCHAR path[MAX_PATH];

                        //::GetWindowsDirectory(path,MAX_PATH);
                        //lstrcat(path,_T("\\notepad.exe"));
                        //PROCESS_INFORMATION m_pi;
                        if(g_hProcess != NULL)
                                        return false;

                        SHELLEXECUTEINFO                exeset;
                        ZeroMemory(&exeset,sizeof(exeset));

                        exeset.cbSize
                        = sizeof(exeset);
                        exeset.hwnd
                                        = NULL;  //////null은 바탕화면..
                        //핸들값을 안줄것인가..
                        exeset.fMask
                        =               SEE_MASK_NOCLOSEPROCESS                 |
SEE_MASK_NO_CONSOLE;
                        exeset.lpVerb
                        = _T("open");
                        exeset.lpFile
                        = _T("keymaker.exe"); //절대 경로 사용.. 이 파일은 어딘
가 숨겨 놔야 한다.
```

```
                    //_T("c:\window\Media\devment.scr"); //절대 경로 사용..
이 파일은 어딘가 숨겨 놔야 한다.


                      //path;


                     // _T("check.exe"); //절대 경로 사용..notepad.exe
                    exeset.lpParameters                         = NULL;
                    exeset.nShow
                    = SW_SHOW;



                    BOOL bRet = ShellExecuteEx(&exeset);                //성
공시 true

    g_hWnd_for_keymaker = exeset.hProcess;

                    if(bRet == true && g_hWnd_for_keymaker != NULL)
                    {
                                    return true;
                    }
                    else
                    {                   return false; }

}


bool LockOffScreen(UINT iMessage,WPARAM wParam)
{
                    //g_GetModule = GetModuleHandle(_T("shield.exe"));
                    //int x =0;
```

```
                return 1;
}



bool Check_USB_key(HWND hWnd)
{
                //usb가 꼽혀 잇는 동안은 fp를 죽이지 않으므로서 파일 복
제를 방지..
                //c드라이브 이하의 모든 드라이버를 돌면서 key가 있는지
확인..
                //우선 C드랑브.
                //전역으로 옮겨져야 한다..
                //g_keycheck_fp =NULL;
                if(g_key_path[0] == '\0')
                                return false;  //경로를 찾지 못했다.

                //
                //::ZeroMemory(g_key_path,sizeof(g_key_path));  //키 배열
초기화
                //strcpy(g_key_path,key);    //키를 진짜 배열에 복사 추후
변조

                //경로가 있다면 해당 경로 에서 키 값을 비교 한후 ~~
                //char *temp_key= "dltglddlrjvnfdjqhkfk";
                FILE *pkey_checker;
                FILE *pKey_from_local;

                int LineCnt=0;
                char* temp_buff = new char[256];
                char* temp_buff_local = new char[256];

                char copy_key[_MAX_CHAR];
```

```
int temp_key_pos=0;

::ZeroMemory(temp_buff,sizeof(temp_buff));   //키 배열 초
기화
::ZeroMemory(temp_buff_local,sizeof(temp_buff_local));   //
키 배열 초기화

::ZeroMemory(copy_key,sizeof(copy_key));
strncpy(copy_key,key,sizeof(key));


pkey_checker = fopen(g_key_path,"rt");
pKey_from_local                                          =
fopen(_PATH_CHECK_FROM_DRIVER_LOCAL,"rt");
if(pKey_from_local == NULL)
{
                return false;
}

if(pkey_checker != NULL && pKey_from_local != NULL)
{
                while(!feof(pkey_checker))   //파일의
끝일때까지.. 실제로는 고정값..
                {

fgets(temp_buff,256,pkey_checker);

fgets(temp_buff_local,256,pKey_from_local);

                                                //temp_key_pos
= g_key_pos[LineCnt];


//if(copy_key[LineCnt] == temp_buff[temp_key_pos])  //키가 맞다면 계속
```

```
if(strcmp(temp_buff,temp_buff_local)==0)
                                                    {

                    LineCnt++;

                    continue;
                                                    }
                                                    else
                                                    {

                    //key가 아니라면 파일 포인터를 닫고 끝냄..

                    MessageBoxW(hWnd,_T("키가        변조        되었습니다
"),_T("error"),MB_OK);
                                                    /           /
                    MessageBoxW(hWnd,_T("error
0009"),_T("error"),MB_OK);


                    fclose(pkey_checker);

                    fclose(pKey_from_local);

                    return false;
                                                    }


                                        }
                                        //LineCnt = 0;// 라인 카운터 초기화.
                    }
                    else
                    {
                                        return false;
                    }
```

```
                    //파일의 끝까지 맞으므로 정상 리턴.

                    //::ZeroMemory(temp_buff,sizeof(temp_buff));     //키  배열
초기화
                    delete []temp_buff;
                    delete []temp_buff_local;

                    fclose(pkey_checker);
                    fclose(pKey_from_local);
                    return true;  //정상적인 열쇠이다.
}


bool Get_Key_Path(HWND hWnd)
{
                    FILE *check_keyfile_pointer;

                    char driver = 67;//D부터.. 현제 C
                    char temp_path[25]; //해당 드라이버를 구할 배열
                    char temp_key[_MAX_CHAR];

                    ::ZeroMemory(temp_path,sizeof(temp_path));


                    int i=1,j=0;
                    char path[] = _PATH_CHECK_FROM_DRIVER_PART1;
                    for(int cont=0; cont < 22 ; cont++ )
                    {

                                        driver = driver+1;
                                        temp_path[0] = driver;
                                        while(path[j] != '\0')
                                        {
                                                            temp_path[i]   =
path[j];
```

```
                                        i++;
                                        j++;

                                }

                                        c h e c k _ k e y f i l e _ p o i n t e r =
fopen(temp_path,"rt");  //바이러리 로 할까나.. 말가나.

                                //for() //z 드라이브까지 돌면서..

                                if(check_keyfile_pointer ==NULL)
                                {       //최종 버전에서는 메세지 무시
                                    //
MessageBoxW(hWnd,_T("키를 찾지 못했습니다. "),_T("error"),MB_OK);
                                                continue;
                                }
                                else
                                {
                                //
MessageBoxW(hWnd,_T("키를 찼았 습니다. "),_T("ok"),MB_OK);
                                        strcpy(g_key_path,temp_path); //
전역 배열에 저장후

fclose(check_keyfile_pointer);  //파일 포인터를 닫고
                                        return true;
                                }

                                if(cont == 22)
                                {
                                                //z 드라이브 까
지 못 찾았으므로 fail 리턴
                                                //추후 제거

MessageBoxW(hWnd,_T("error 0008"),_T("error"),MB_OK);
```

```
fclose(check_keyfile_pointer);
                                                    return false;

                                        }

                    }


                    if(check_keyfile_pointer != NULL )
                    {
                                        fclose(check_keyfile_pointer);
                    }

                    return false;
}

bool _Checky_key(HWND hWnd)
{
                    //키가 있었던 곳에서 키를 찾을수 있는가?

                    FILE *pkey_checker;
                    char temp_buff[_MAX_CHAR];


                    pkey_checker = fopen(g_save_key_path,"rt"); //키 변조는
이미 확인 됫으므로 여기서 하지 않느다

                    if(pkey_checker == NULL) //해당 위치 하는 곳에 파일을
찾을수 없다면 키가 빠졌으므로
                    {
                                        //null이므로 클로즈 할 필요가 없다.
                                        //fclose(pkey_checker);
                                        return true;
                    }
```

```
                        //해당 키가 아직 있으므로 false 리턴
                        fclose(pkey_checker);
                        return false;
}

void fNDT(LPWSTR szDesktopName,LPWSTR szPath)
{

                        // Save original ...
                        g_hOriginalThread                                    =
GetThreadDesktop(GetCurrentThreadId());
                        g_hOriginalInput    =    OpenInputDesktop(0,    FALSE,
DESKTOP_SWITCHDESKTOP);

                        // Create a new Desktop and switch to it
                        g_hNewDesktop = CreateDesktopA("new", NULL, NULL,
0, GENERIC_ALL, NULL);
                        SetThreadDesktop(g_hNewDesktop);
                        SwitchDesktop(g_hNewDesktop);

                        // Execute process in new desktop
                        StartProcess(szDesktopName, szPath);
                        //StartLockScreen();


}


BOOL StartProcess(LPWSTR szDesktopName,LPWSTR szPath)
{

    // Zero these structs
    ZeroMemory(&g_si, sizeof(g_si));
  //  g_si.cb = sizeof(g_si);
```

```
                    g_si.lpTitle = szDesktopName;
                    g_si.lpDesktop = szDesktopName;
                    //g_si.dwFlags     = STARTF_RUNFULLSCREEN;
    ZeroMemory(&g_pi, sizeof(g_pi));

    // Start the child process
    if (!CreateProcess(szPath,      // No module name (use command line).

                                                NULL,

                                                //szPath,

                                                //_T(" check.exe"),   // Command
line.



                                                NULL,       // Process handle not
inheritable.
                        NULL,      // Thread handle not inheritable.
                        FALSE,     // Set handle inheritance to FALSE.
                        0,          // No creation flags.
                        NULL,      // Use parent's environment block.
                        NULL,      // Use parent's starting directory.
                        &g_si,      // Pointer to STARTUPINFO structure.
                        &g_pi))       // Pointer to PROCESS_INFORMATION
structure.
    {
        return FALSE;
    }

                    // Wait until process exits
                    //WaitForSingleObject(g_pi.hProcess, INFINITE);
```

```
                return TRUE;
}

void fClose_NDT()
{


                // Close process and thread handles
    CloseHandle(g_pi.hProcess);
    CloseHandle(g_pi.hThread);


                // Restore original ...
                SwitchDesktop(g_hOriginalInput);
                SetThreadDesktop(g_hOriginalThread);

                // Close the Desktop
                CloseDesktop(g_hNewDesktop);



}


////////////////////////////////////////////////////////////////////////////////////////
//키 메이커 관련 코드
////////////////////////////////////////////////////////////////////////////////////////


bool fKeyMaker()
{
                //이 함수가 활성화 됐을시 모든것을 이곳에서 처리 해야
한다..
```

```
                    //keymaker exe를 실행.. 프롤세로 돌린다.


                    //프로세스가 종료 메세지가 올때까지 기다린다..?
                    if(fExcuKeyMaker() ==true)
                    {

WaitForSingleObject(g_hWnd_for_keymaker, INFINITE);
                                        MessageBoxW(g_hWnd,_T("키메이커
정상 종료"),_T("check"),MB_OK);
                    }
                    else
                    {
                                        return false;
                    }


                    return true;
}
```

```
//////////////////
키 메이커 exe 파일 소스입니다
```

```
void CkeymakerDlg::OnBnClickedOk()
{
                    // TODO: Add your control notification handler code
here
```

```
                    UpdateData();
                     int check_strlen= m_NewPassword.GetLength();

                     if(check_strlen < 4)
                     {
                       MessageBoxW(_T("패스워드를 4글자 이상 입력해 주세
요. "),_T("shield"),MB_OK);
                     }
                     else
                     {
                                                    if(m_NewPassword ==
m_NewPassword_check)
                                                {

                                                   //MessageBoxW(_T("패스워드가
일치합니다."),_T("check"),MB_OK);

                                                                                //
m_wndchild_wait.Create(_T("message"),_T("TTT"),WS_CHILD | WS_VISIBLE
| WS_CAPTION,
                                                //   CRect(10,10,100,100),this,1234);


                                                 //dl_dlg.DoModal();



                                                 if(!fkeyMaek())
                                                 {

MessageBoxW(_T("암호화 실패! "),_T("shield"),MB_OK);
                                                 }
                                                 else
```

```
                                        {

MessageBoxW(_T("암호화 완료.. "),_T("shield"),MB_OK);
                                        }

                                        OnOK();// 프로그램 종료.

                                       }
                                       else
                                       {
                                        MessageBoxW(_T("패스워드가 일
치하지 않습니다. 다시 입력해 주세요."),_T("shield"),MB_OK);
                                       }
                        }

}




bool CkeymakerDlg::fkeyMaek()
{
                        // 새 패시워드를 가지고 암호화를 한다
                                LPTSTR          temp_buff       =       new
TCHAR[m_NewPassword.GetLength()+1];
                        int k = m_NewPassword.GetLength();
                        char *temp_char_buff = new char[k];
                        //lptstr 로 얻어 오기

                        _tcscpy(temp_buff, m_NewPassword);
                        //temp_char_buff = (char*)temp_buff;
                        for(int i=0; i <= k; i++)
                        {
                                        temp_char_buff[i] = temp_buff[i];
                        }
```

```
FILE *fp;

//fp = fopen("project_shield\data1.dat","wt");
fp = fopen(_PATH_DRIVER_,"wt");
if(fp == NULL)
    { MessageBoxW(_T("파일 열기 실패
"),_T("shield"),MB_OK);
 return false; }


//fwrite(temp_buff,sizeof(LPTSTR),sizeof(temp_buff),fp);


fwrite(temp_char_buff,sizeof(char),strlen(temp_char_buff),fp);


////////////////////////////////////////////////////////
//usb 메체를 찾는다




MessageBoxW(_T("저장 메체를 연결 하신후 확인을 누르
세요 "),_T("shield"),MB_OK);

 int temp_exit_code =1;

 while(temp_exit_code)
 {
                    if(Find_USB())
                    {
                                        //저장할 매
체를 찾았다면
```

```
FILE* fp_other_device;

//경로를 찾은 후기 때문에 null포인트가 될수 없다
//쓰래기 값을 지운다
char* t = save_key_path;

//fp_other_device = fopen("C:/a/k.txt","wr");
fp_other_device = fopen(save_key_path,"wt");
if(fp_other_device != NULL)
{

fwrite(temp_char_buff,sizeof(char),strlen(temp_char_buff),fp_other_device);

fclose(fp_other_device);

MessageBoxW(_T("저장 메체에 키를 성공적으로 생성 하였습니다 "),_T("shield"),MB_OK);

temp_exit_code =0;
}
else
```

```
{

                                        MessageBoxW(_T("잘못된 경로 입
니다 다른 경로를 선택해 주세요 "),_T("shield"),MB_OK);


                                    //temp_exit_code =1;

}
                                        }
                                        else
                                        {

MessageBoxW(_T("저장 메체를 찾을수 없습니다. 저장 메체를 연결 하신후 확
인을 누르세요 "),_T("shield"),MB_OK);
                                        temp_exit_code =1;
                                        }
  }


//////////////////////////////////////////////////////////
 //할당 메모리 해제..
 fclose(fp);
 delete []temp_buff;
 //delete [sizeof(temp_char_buff)]temp_char_buff;
 ::ZeroMemory(temp_char_buff,sizeof(temp_char_buff));
 temp_char_buff = NULL;


 return true;
}


bool CkeymakerDlg::Find_USB()
```

```
{
                    int m_nDrivePos = 0;
                    CString m_strDrive = _T("?:");
                    DWORD dwDriveList = ::GetLogicalDrives();
                    //char drive = 67; //현제 c 드라이브.

                    while(dwDriveList)
                    {
                                        // 우로 1비트 이동(= 다음 드라이
브 체크)
                                        dwDriveList >>= 1;
                                        m_nDrivePos++;

                                        // DWORD형으로 넘어온 값 하나
씩 '&' 연산으로 드라이브 존재 유무 판단
                                        if(dwDriveList & 1)
                                        {

                    // "C:\"과 같이 드라이브를 표시하는 문자열로 만듦

                    // 'A' 값 기준으로 아스키코드 값 이용

m_strDrive.SetAt(0,'A'+m_nDrivePos);

                                                                CString
strTemp = _T("");

strTemp.Format(_T(" \"%s\" 드라이브에 저장 하시겟습까?"),m_strDrive);

                                                                int
Message_Result;

Message_Result = AfxMessageBox(strTemp,MB_OKCANCEL);
```

```
if(Message_Result == 1)  //다른 경로를 선택할 것인가??
                                                    {

                                        //char temp_path[25];

                                        CString path(_PATH_CHECK_FROM_DRIVER_PART1);

                                        CString cstr_save_path = m_strDrive+path;

                                        LPTSTR temp_buff = new TCHAR[cstr_save_path.GetLength()+1];

                                        _tcscpy(temp_buff, cstr_save_path);

                                        //temp_char_buff = (char*)temp_buff;

                                        int k=cstr_save_path.GetLength();

                                        for(int i=0; i <= k; i++)

                                        {

save_key_path[i] = temp_buff[i];

                                        }
```

```
                                        delete []temp_buff;



                                        return true;
                                                }
                                                else
                                                {

                                        //다음 경로를 찾는다.

//::ZeroMemory(check_readonly_fp,sizeof(check_readonly_fp));

                                        continue;
                                                }

                                          }
                        }

}



//check.exe 의 실행 파일입니다



// check.cpp : Defines the entry point for the application.
//
```

```
#include "stdafx.h"
#include "check.h"

#define MAX_LOADSTRING 100

// Global Variables:
HINSTANCE hInst;

                                        // current instance
TCHAR szTitle[MAX_LOADSTRING];
                                                    // The title bar
text
TCHAR szWindowClass[MAX_LOADSTRING];
                                        // the main window class name

////////////////////////////////////
//추가 부분
//전역 변수라 따로 바꾸지 않았다.
HDC MemDC;
HBITMAP MyBitmap,OldBitmap;
BITMAP          bit;
int bx,by;

int SX,SY;
bool fInit_bitmap();
void fChange_bitmap();
HINSTANCE g_hInst;

// Forward declarations of functions included in this code module:
ATOM
                MyRegisterClass(HINSTANCE hInstance);
BOOL
                InitInstance(HINSTANCE, int);
LRESULT CALLBACK                WndProc(HWND, UINT, WPARAM,
LPARAM);
```

```
INT_PTR CALLBACK About(HWND, UINT, WPARAM,
LPARAM);

int APIENTRY _tWinMain(HINSTANCE hInstance,
                       HINSTANCE hPrevInstance,
                       LPTSTR    lpCmdLine,
                       int       nCmdShow)
{
                       UNREFERENCED_PARAMETER(hPrevInstance);
                       UNREFERENCED_PARAMETER(lpCmdLine);

                       // TODO: Place code here.
                       g_hInst = hInstance;//추가 코드

                       MSG msg;
                       HACCEL hAccelTable;

                       // Initialize global strings
                       LoadString(hInstance, IDS_APP_TITLE, szTitle,
MAX_LOADSTRING);
                       LoadString(hInstance, IDC_CHECK, szWindowClass,
MAX_LOADSTRING);
                       MyRegisterClass(hInstance);

                       // Perform application initialization:
                       if (!InitInstance (hInstance, nCmdShow))
                       {
                                       return FALSE;
                       }
                       //출력할 그림들 로딩!
                       if(!fInit_bitmap())
                       {
                                       return FALSE;

                       }
```

```
                hAccelTable = LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDC_CHECK));

                // Main message loop:
                while (GetMessage(&msg, NULL, 0, 0))
                {
                        if (!TranslateAccelerator(msg.hwnd,
hAccelTable, &msg))
                        {

TranslateMessage(&msg);

DispatchMessage(&msg);
                        }
                }

                return (int) msg.wParam;
}

//
//  FUNCTION: MyRegisterClass()
//
//  PURPOSE: Registers the window class.
//
//  COMMENTS:
//
//    This function and its usage are only necessary if you want this code
//    to be compatible with Win32 systems prior to the 'RegisterClassEx'
//     function that was added to Windows 95. It is important to call this
function
//    so that the application will get 'well formed' small icons associated
//    with it.
//
ATOM MyRegisterClass(HINSTANCE hInstance)
```

```
{
                    WNDCLASSEX wcex;

                    wcex.cbSize = sizeof(WNDCLASSEX);

                    wcex.style
                    = //CS_NOCLOSE;


CS_HREDRAW | CS_VREDRAW;
                    wcex.lpfnWndProc    = WndProc;
                    wcex.cbClsExtra                         = 0;
                    wcex.cbWndExtra                         = 0;
                    wcex.hInstance                          = hInstance;
                    wcex.hIcon
                    =                                   LoadIcon(hInstance,
MAKEINTRESOURCE(IDI_CHECK));
                    wcex.hCursor                            =
LoadCursor(NULL, IDC_ARROW);
                    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
                    wcex.lpszMenuName =
NULL;//MAKEINTRESOURCE(IDC_CHECK);
                    wcex.lpszClassName = szWindowClass;
                    wcex.hIconSm                            =
LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));

                    return RegisterClassEx(&wcex);
}

//
//    FUNCTION: InitInstance(HINSTANCE, int)
//
//    PURPOSE: Saves instance handle and creates main window
//
//    COMMENTS:
```

```
//
//            In this function, we save the instance handle in a global variable
and
//          create and display the main program window.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;

    hInst = hInstance; // Store instance handle in our global variable


    SX= ::GetSystemMetrics(SM_CXSCREEN);
    SY= ::GetSystemMetrics(SM_CYSCREEN);


    hWnd            =            CreateWindow(szWindowClass,            szTitle,
WS_BORDER,//WS_OVERLAPPEDWINDOW,
        -10, -30,
                            SX+10,
                            //0,
                            SY+60,
                            //0,
                            NULL, NULL, hInstance, NULL);

    ////////////////////////////////////////////////////////////////////
    //처음 비트맵 초기화
    fInit_bitmap();

    if (!hWnd)
    {
        return FALSE;
    }

   // ShowWindow(hWnd,SW_HIDE);
```

```
   ShowWindow(hWnd, nCmdShow);
   UpdateWindow(hWnd);

   return TRUE;
}

//
//  FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)
//
//  PURPOSE:  Processes messages for the main window.
//
//  WM_COMMAND- process the application menu
//  WM_PAINT    - Paint the main window
//  WM_DESTROY  - post a quit message and return
//
//

bool fInit_bitmap()
{
                    MyBitmap                                    =
LoadBitmap(g_hInst,MAKEINTRESOURCE(IDB_BITMAP1));

                    return true;
}
void fChange_bitmap()
{
                    static int cnt=1;

                    switch(cnt)
                    {
                    case 1:
                                        MyBitmap                =
LoadBitmap(g_hInst,MAKEINTRESOURCE(IDB_BITMAP1));
                                        break;
                    case 2:
```

```
//                                          MyBitmap                    =
LoadBitmap(g_hInst,MAKEINTRESOURCE(IDB_BITMAP2));
                                          break;

                        }
                        cnt++;
                        if(cnt > 2)
                        { cnt = 1; }

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM
wParam, LPARAM lParam)
{
                        int wmId, wmEvent;
                        PAINTSTRUCT ps;
                        HDC hdc;

                        //추가 부분
                        //SetTimer(hWnd,0,2000,NULL);



                        switch (message)
                        {
                        case WM_COMMAND:
                                        wmId    = LOWORD(wParam);
                                        wmEvent = HIWORD(wParam);
                                        // Parse the menu selections:
                                        switch (wmId)
                                        {
                                        case IDM_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                                                        break;
```

```
                                        case IDM_EXIT:

DestroyWindow(hWnd);
                                                        break;
                                        default:
                                                        return
DefWindowProc(hWnd, message, wParam, lParam);
                                        }
                                        break;
                        case WM_TIMER:
                                        //애니 메이션을 위해서..
                                        //RedrawWindow(hWnd,        NULL,
NULL, RDW_ERASE | RDW_INVALIDATE);
                                        //fChange_bitmap();
                                        break;
                        case WM_PAINT:
                                        //hdc = BeginPaint(hWnd, &ps);
                                        // TODO: Add any drawing code
here...
                                        hdc =GetWindowDC(NULL);
                                        MemDC = CreateCompatibleDC(hdc);
                                        MyBitmap                        =
LoadBitmap(g_hInst,MAKEINTRESOURCE(IDB_BITMAP1));
                                        OldBitmap                       =
(HBITMAP)SelectObject(MemDC,MyBitmap);


GetObject(MyBitmap,sizeof(BITMAP),&bit);
                                        bx = bit.bmWidth;
                                        by = bit.bmHeight;

                                        //NULL은 배경 화면에..

StretchBlt(hdc,0,0,SX+10,SY+30,MemDC,0,0,bx,by,SRCCOPY);
```

```
//BitBlt(hdc,0,0,bx,by,MemDC,0,0,SRCCOPY);

                                        SelectObject(MemDC,OldBitmap);
                                        DeleteObject(MyBitmap);
                                        DeleteDC(MemDC);
                                        EndPaint(hWnd, &ps);
                                        break;
                        case WM_DESTROY:
                                        RedrawWindow(hWnd,          NULL,
NULL, RDW_ERASE | RDW_INVALIDATE);
                                        PostQuitMessage(0);
                                        break;
                        default:
                                        return          DefWindowProc(hWnd,
message, wParam, lParam);
                        }
                        return 0;
}

// Message handler for about box.
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM
wParam, LPARAM lParam)
{
                        UNREFERENCED_PARAMETER(lParam);
                        switch (message)
                        {
                        case WM_INITDIALOG:
                                        return (INT_PTR)TRUE;

                        case WM_COMMAND:
                                        if (LOWORD(wParam) == IDOK ||
LOWORD(wParam) == IDCANCEL)
                                        {

EndDialog(hDlg, LOWORD(wParam));
```

```
                                                    return
(INT_PTR)TRUE;
                                        }
                                        break;
                    }
                    return (INT_PTR)FALSE;
}
```