**软件注册码验证动态库封装**

软件注册码生成思路：

1. 获取CPUID
2. 获取磁盘序列号
3. 获取MAC地址
4. 将以上三种信息进行MD5加密，得到32个字符，由这32字符做私有加密处理

最终得到唯一的序列号

软件注册码验证过程跟上述方法一致。

下面是验证过程动态库封装及使用：

新建MFC dll工程，为CVerifyCodeApp类添加成员函数，

CVerifyCodeApp();

CString GetCpuNum();

CString GetDiskNum();

CString GetRegCode(CString szSerialInfo);

bool VerifyCode();

public:

CMd5Encode m\_md5Encode;

以上函数的定义如下：

CString CVerifyCodeApp::GetCpuNum()

{

//获取CPU序列号

unsigned long s1, s2;

CString szCpuID1(\_T("")), szCpuID2(\_T(""));

\_\_asm {

mov eax, 01h

xor edx, edx

cpuid

mov s1, edx

mov s2, eax

}

szCpuID1.Format(\_T("%08X%08X"), s1, s2);

\_\_asm{

mov eax, 03h

xor ecx, ecx

xor edx, edx

cpuid

mov s1, edx

mov s2, ecx

}

szCpuID2.Format(\_T("%08X%08X"), s1, s2);

//拼接结果

CString szCpuID = szCpuID1;

return szCpuID;

}

CString CVerifyCodeApp::GetDiskNum()

{

DWORD dwSerial;

char cDiskName[128];

char cFileSys[128];

CString szDiskID(\_T(""));

::GetVolumeInformation(\_T("c:\\"), //磁盘根路径

(LPWSTR)cDiskName, //用于装载磁盘名的缓冲区

128, //缓冲区大小

&dwSerial, //装载磁盘序列号的变量

0, //装载文件名每一部分的长度

0, //标志位

(LPWSTR)cFileSys, //装载文件系统的名称

128 //缓冲区大小

);

szDiskID.Format(\_T("%08X"), dwSerial);

return szDiskID;

}

CString CVerifyCodeApp::GetRegCode(CString szSerialInfo)

{

int nIndex, nPostion;

CString szRes(\_T(""));

//定义一个秘钥数组

CString szCode[16] = {

\_T("ah"), \_T("tm"), \_T("ib"), \_T("nw"), \_T("rt"), \_T("vx"), \_T("zc"), \_T("gf"),

\_T("pn"), \_T("xq"), \_T("fc"), \_T("oj"), \_T("wm"), \_T("eq"), \_T("np"), \_T("qw")

};

for(nIndex=0; nIndex < szSerialInfo.GetLength(); nIndex++)

{

char cTemp = szSerialInfo[nIndex];

if(cTemp >= 'a' && cTemp <= 'z')

{

nPostion = cTemp - 'a' + 10;

}

else

{

nPostion = cTemp - '0';

}

szRes = szRes + szCode[nPostion];

}

szRes.MakeUpper();

return szRes;

}

bool CVerifyCodeApp::VerifyCode()

{

CString szResDir(\_T("")), szKeyID(\_T("")), szResSN(\_T(""));;

char szBuf[1025] = {0};

GetModuleFileName(NULL, (LPWCH)szBuf, sizeof(szBuf));

szResDir.Format(\_T("%s"), szBuf);

szResDir = szResDir.Left(szResDir.ReverseFind('\\'));

szKeyID = GetCpuNum() + GetDiskNum();

string sKeyID1, sSerialInfo, szTemp;

sKeyID1 = CStringA(szKeyID);

sSerialInfo = m\_md5Encode.Encode(sKeyID1);

szTemp = szTemp + sSerialInfo[0]+sSerialInfo[1]+sSerialInfo[2]+sSerialInfo[3]+sSerialInfo[7]+sSerialInfo[15]+

sSerialInfo[31]+sSerialInfo[5]+sSerialInfo[11]+sSerialInfo[22]+sSerialInfo[30]+sSerialInfo[9]+sSerialInfo[10]+sSerialInfo[17];

szResSN = GetRegCode((CString)szTemp.c\_str());

CStdioFile clsFile;

CFileFind clsFileFind;

CString szLine(\_T(""));

if(!clsFileFind.FindFile(szResDir+\_T("\\license.txt")))

{

AfxMessageBox(\_T("未找到认证文件license.txt！"));

return false;

}

else

{

if(!clsFile.Open(szResDir+\_T("\\license.txt"), CFile::modeRead))

{

AfxMessageBox(\_T("打开认证文件license.txt失败！"));

return false;

}

else

{

while(clsFile.ReadString(szLine))

{

szLine.Remove(' ');

if(szLine.Left(3) == \_T("sn:"))

{

szLine = szLine.Right(szLine.GetLength()-3);

break;

}

}

if(szLine != szResSN)

{

AfxMessageBox(\_T("认证失败！"));

return false;

}

}

}

return true;

}

将md5encode.h md5encode.cpp文件添加入工程中，在VerifyCode.def文件中添加 VerifyCode @1;

编译得到 VerifyCode.dll

动态库的加载过程如下：

在应用工程的App类头文件中加入 如下连个变量

HMODULE m\_handle;

fVerifyCode m\_fVerifyCode;

添加一个预处理：typedef bool (\_\_stdcall \*fVerifyCode)();

在App类的InitInstance()函数中添加如下代码

{

m\_handle = LoadLibrary(\_T("VerifyCode.dll"));

if(!m\_handle)

{

AfxMessageBox(\_T("加载动态库失败！"));

exit(0);

}

m\_fVerifyCode = (fVerifyCode)GetProcAddress(m\_handle, "VerifyCode");

if(!m\_fVerifyCode)

{

AfxMessageBox(\_T("获取函数地址失败！"));

return FALSE;

}

if(!m\_fVerifyCode())

{

return FALSE;

}

}

即可验证