```
  / * 取磁盘缓冲区首地址 * /
r.h.ah=0x2f;'
int86(0x21,&r,&r);
bx=r.x.bx;
segread(&s);
ds=s.es;
for (i=0;i<512;i++)
        buff[i]=peekb(ds,bx+i);

r.x.bx=bx;
s.ds=ds;
r.x.ax=al;
r.x.cx=1;
r.x.dx=0;
int86x(0x25,&r,&r,&s);]
  / * 计算目录首地址,总扇区数 * /
if (r.x,cflag==1j)
    {printf("\ndu pan cuo");
    x=0;
    n=-5;
    }
else
    {x=peek(ds,bx+22) * 2+1;
    n=peek(ds,bx+7) * 32 / 512;
    }
  / 读目录表 * /
for(i=0;i<n;i++)
    {r.x.bx=bx;
     s.ds=ds;
     r.x.ax=al;
     r.x.cx=1;
     r.x.dx=x+i;
     int86x(0x25,&r,&r,&s);
     for(j=0;j<512;j=j+32)
       {for(k=0;k<11;k++)
            fil2[k]=peekb(ds,bx+j+k);
        / 文件找到,则加密或解密 * /
       if (strcmp(fil1,fil2)==-32)
         {if (peek(ds,bx+j+26)!=0)
            {poke(ds,bx+j+12,peek(ds,bx+j+26));
             poke(ds,bx+j+26,0);
             pokeb(ds,bx+j+15,peekb(ds,bx+j+11));
             pokeb(ds,bx+j+11,39);
             printf(" / nyi jia mi");
             }
          else
            {poke(ds,bx+j+26,peek(ds,bx+j+12);
             poke(ds,bx+j+12,0);
             pokeb(ds,bx+j+11,peekb(ds,bx+j+15));
             pokeb(ds,bx+j+15,0);
             printf(" / nyi jie mi");
             }
          r.x.ax=al;
          r.x.cx=1;
          r.x.dx=x=i;
          r.x.bx=bx;
          s.ds=ds;
          int86x(0x26,&r,&r,&s);
          n=-1;
          break;
          }
       }
    }
if(n!=-1 && n!=-5)
  printf(" / nwen jian mei zhao dao");
for(i=0;i<512;i++)
  pokeb(ds,bx+i,buff[i]);
}
```

"m 序列"及其在加密领域的应用

王咏武　　(无锡市工商银行电脑科)

摘要:微机的硬口令是以密文形式存储在 CMOS 中,并且多采用"m 序列"进行加密,本文即以微机硬口令为例来说明"m 序列"以及利用它进行加、解密的方法。

一、CMOS 的组成及其接口

现有微机都采用 CMOS 来存储配置信息,CMOS 的长度一般为 64 或 128 字节,其读写接口是 70H 和 71H,其中 70H 称为索引端口,71H 称为数据端口,读写 CMOS 中某字节的值时,只须在 70H 口输出该字节的偏移,然后在 71H 口即可读写该字节的值。硬口令存储在偏移 37H～3DH 共七个字节中,多数微机采用"m 序列"进行加密。例如下达代码可以读出口令的第一个字节。

```
char val;
outp(0x70,0x37);
val=inp(0x71);
```

二、"m 序列"及其生成

"m 序列"是用带线性反馈的移位寄存器产生的周期最长的一种序列。对于 n 位的移位寄存器来说,其周期应为 2n−1。例如:n=3,设其初始状态为 001,经过 8 次移位其状态依次为:001、100、010、101、111、011、001,又回到了初始状态,并且循环往复,所以其周期为 7,一个周期中遍历了 001～111 中的所有状态,这样的序列就称作"m 序列"。

一个一般的线性反馈移位寄存器如图 1 所示,其中 Co 和 Cu 必为 1,当某位 Ci 为 1 时,表示存在该条反馈线,当 Ci 为 0 时,则不存在该条反馈线,所有的加法都为模 2 相加。

由于 C_0～C_n 的取值决定了该移位寄存器的结构,故 Ci 是一个很重要的参量,因此将它用下面的多项式表示:

$$f(x)=C_0+C_1x+C_2x^2+\cdots\cdots+C_nx^n=\sum_{i=0}^{n}C_ix^i$$

这一多项式称作特征多项式,x 的值并无实际意义,仅是用来指明 Ci 的值。

可以证明,一个能够生成"m 序列"的移位寄存器,其特征多项式必须满足下列条件:

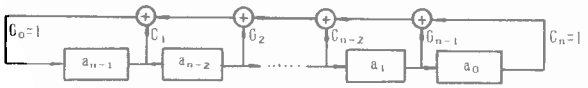

图 1

1.f(x)必须是即约多项式,即不能分解因子的多项式。

2.f(x)是(x^p+1)的一个因子,其中 $p=2^n-1$。

3.对于所有的 $q<p$,f(x)不是(x^q+1)的因子。

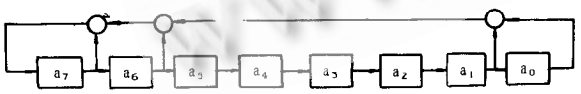

图 2

可见,如何求出其特征多项式是构造"m 序列"生成器的关键,但是这一工作非常困难。幸运的是,前人已经将这些特征多项式列成表备查,其中 n=8 的特征多项式最为常用,硬口令加密时所用的特征多项式是 $x^8+x^7+x^2+x+1$,所构成的移位寄存器如图 2 所示,利用该移位寄存器就可以生成 8 位的"m 序列"。

三、如何利用"m 序列"进行加、解密

CMOS 中的口令以密文形式存储在 37H～3DH 中,共七个字节,其明文最长为六个字节,加密方法如下:

取一个任意值,作为上述移位寄存器的初始状态,并存入 37H 中,取口令的第一个字符 P_1,将该移位寄存器移位 P_1,取其状态存入 38H 中,取口令的第二个字节)P_2,将移位寄存器再移位 P_2 步,取其状态存入 39H 中,重复上述步骤直至口令的六个字符都处理完毕。

解密时相反,取 37H 中的值作为移位寄存器的初始值,将其移位直至其状态等于 38H 中的值,所用步数即是口令中的第二个字符,重复以上步骤,直至口令的六个字节都已解码完毕。

自己编制加密软件时,还可以通过改变初始状态和移位步数的计算方法来进一步提高算法的保密性。

四、结束语

文后所附的程序是 CMOS 口令的解码程序,运行后即可获得当前机器的口令。因有部分机器不用"m 序列"加密法加密,只是将口令字符的扫描码直接存入 38H～3DH 中,所以本程序也包含处理这种加密方法的代码。该程序在 BORLAND C++环境下用小模式编译通过。

"m 序列"加密法运用于很多场合,具有较高的保密性,有一定的实用价值。

```
/ * * * * * * * * * * * * * * * *
 * passwd.c                      *
 * Copywrite by Wang Yong Wu *
 * 1995.6.28                     *
 * * * * * * * * * * * * * * * * /
#include < conio.h >
#include < stdlib.h >
#include < stdio.h >
#include < ctype.h >
#define PORTINDEX 0x70 / / cmos index port
#define PORTDATA 0x71 / / cmos data port
#define BEGIN 0x37 / / password offset
void m  unpass(void); / / M queue unpass
void scan  unpass(void); / / scan code unpass
char readbuf[7];
char passwd[7];
char scan[256] = {0,0,'1','2','3','4','5','6','7','8','9','0',
```

```
'-', '=', 0, 0, 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I',
'O', 'P', '[', ']', 0, 0, 0, 'A', 'S', 'D', 'F', 'G', 'H',
'J', 'K', 'L', ';', '\'', '`', 0, '\\', 'Z', 'X', 'C',
'V', 'B', 'N', 'M', ',', '.', '/', 0, 0, 0};
main()
{
int i;
printf("Password 1.0! / nCopywrite by Wang Yong Wu! / n / n");
for (i=0;i<7;i++)
{outp(PORTINDEX,(char)i+BEGIN);
redbuf[i]=inp(PORTDATA);
printf("The password number%d is :%x / n",i,readbuf[i]);
} / read from cmos 37h--3dh
printf(" / n");
m unpass();
printf(f" / n");
scan unpass();
printf("OK"! / n");
return 0;
}
void m unpass(void)
{
int i;
unsigned char buf,shift;
readbuf[0] &=0xf0;
for (i=0;i<6;i++)
{if(!readbuf[i+1])break;
passwd[i]=1;buf=readbuf[i];
for(;;)
{shift=((buf & 0x80)>>7)+((buf & 0x40)>>6)
+((buf & 0x02)>>1)+buf & 0x1;
shift=(shift & 0x1)<<7;
buf=(buf>>1)|shift;
if(buf==readbuf[i+1]    passwd[i]='?';
}
printf("IF use M queue; The password is:%s / n",passwd);
}
void scan unpass(void)
{
int i;
for (i=1;i<7;i++)
{if(!readbuf[i] break;
if(!scan[readbuf[i]])passwd[i-1]='?';
else passwd[i-1]=scan[readbuf[i]];
}
printf("IF use scan code;The password is:%s / n",passwd);
}
```

用0磁道格式化清除主引导区新病毒

董翔英　　(海军后勤学院)

摘要:本文介绍一种不能用覆盖法清除的主引导区病毒的解决办法。

我单位两台长城386微机不慎被人安装某一不知名的财务软件,感染了一个未曾见过的病毒,导致硬盘不能启动。采取一般方法,如用软件杀毒法、主引导区覆盖法、物理格式化后重新分区法,均不见效。最后在万般无奈时,采取将硬盘0头0面0道单独格式化后,才消除了病毒。现将该病毒的病毒现象及消除过程介绍给大家,期望在病毒日益猖厥的今天,能够有助于同行在工作中遇到同类病毒袭击时,开拓思路,清除病毒。

一、病毒现象

1.开机系统自检正常,当A驱不插系统盘时,C盘不能启动机器,并且A驱灯亮,发出沉重的读盘声响,持继不断;

2.当A驱插入系统盘后,系统启动成功。能转C盘,并且此后一切读写操作均正常;

3.当启动时按下DEL键试图进入系统设置,则询问口令(原系统设置并无口令检测);

4.用软盘启动后读出主引导扇区内容,发现分区表仍是正确的,但主引导程序已是面目全非。很显然,该机主引导区已被病毒感染;

5.用手中现有的最高版本的杀毒软件,如:CPAV、KILL、SCAN等进行清毒时,均查不出该病毒;

6.当进入DEBUG,试图将好的主引导区写至该机的坏主引导区时,硬盘指示灯不亮,系统不能进行写操作;

7.用FDISK删除所有分区(该机只有一个DOS分区),进行硬盘重新分区,屏幕显示error disk write,操作失败;

8.用DM的自动方式作硬盘物理格式化时,硬盘类