基于C++口令文件加密算法的实现

An Algorithm of Password Encryption Based on C++

黄珍生

Huang Zhensheng

(**广西民族学院计算机与信息科学学院,广西南宁** 530006)

(Coll. of Comp. & Info. Sci., Guangxi Univ. for Nationalities, Nanning, Guangxi, 530006, China)

摘要:分析口令文件的加密思路,提出一种口令文件加密算法。该口令文件加密算法以函数的形式表述,攻击者即使获取口令文件中的已编码口令,也无法对它们进行译码。

关键词:口令文件 加密 算法

中图法分类号:TP311.1 **文献标识码**:A **文章编号**:1002-7378(2005)02-0113-02

Abstract: An algorithm of the password encryption is developed to solve the problem of password to be decrypted easily. This algorithm was presented by using the function forms of C++ language. Password encrypted storage is not decrypted easily by the attacker, even though the password codes in the password files have been obtained.

Key words: password file, encryption, algorithm

利用口令来确认用户身份,是当前最常用的认证技术。为防止用户口令被攻击者获取,登录程序采用了一些安全措施,但用户的用户名、口令和用户权限等信息最终都存放在口令文件中,如果攻击者找到了口令文件并轻易地打开,那么系统的合法用户名、口令等信息将暴露无遗[1]。因此,在口令机制中,口令文件的安全性至关重要。显然,如何保证口令文件的安全性,已成为系统安全性的头等重要问题。为此本文提出一种口令文件加密算法,采取这一加密算法后,即使攻击者获取口令文件中的已编码口令,也无法对它们进行译码,大大提高了系统的安全性。

1 口令文件的加密思路

保证口令文件安全性的最有效方法是利用加密技术对口令文件加密后再保存。对口令文件加密有2种途径,一种是对整个口令文件进行加密处理后保存;另一种是只对口令文件中的一些关键信息项进行加密处理保存[2]。本文主要讨论第二种情况。

我们的加密思路是,选择一个函数来对口令进行加密,该函数 $f(x)$ 选择使其具有这样的特性:在给定了 x 值后,易算出 $f(x)$;然而,如果给定了 $f(x)$ 的值,却不能算出 x 的值。利用 $f(x)$ 函数去编码(即加密)所有的口令,再将加密后的口令存入口令文件。当某用户输入一个口令时,系统利用函数 $f(x)$ 对该口令进行编码,然后将编码(加密)后的口令与存储在口令文件中已编码的口令进行比较,如果两者相匹配,便认为是合法用户。采取这一措施后,即使攻击者能获取口令文件中的已编码口令,也无法对它们进行译码,因而不会影响到系统的安全性。图1给出了加密口令进行验证的方法。

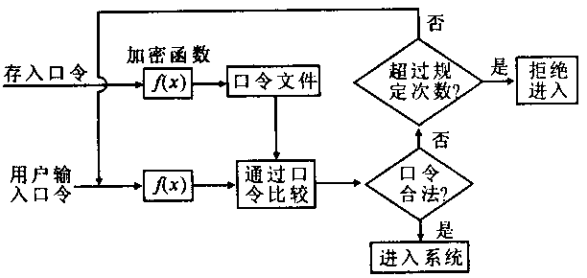

图1 加密口令的验证

2 口令文件的加密算法

下面给出对口令文件的加密算法和基于C++实现方法[3]。

2.1 无第三方认证情况下的加密算法

如果一个系统需要系统管理员提供密钥,则用户须提供口令才能进入系统,其加密算法描述如下:

收稿日期:2005-02-02

作者简介:黄珍生(1965-),男,讲师,主要从事计算机软件研究。

步骤 1 获取用户口令的字符长度 L，系统管理员提供初始密钥 k，令计数器 $i=0$；

步骤 2 将密钥 k 进行一定的算术运算或逻辑运算（如右移一个字节）后再与用户输入口令的第 i 个字符进行一定的算术运算或逻辑运算（如异或操作），所得到的字符再保存在口令文件中；

步骤 3 由刚得到的字符与其所用的密钥 k 再做一定的运算（如相加操作）后形成新的密钥 k；

步骤 4 判断 $i<L$？如果成立，则转步骤 2，否则，转步骤 5；

步骤 5 结束。

上述口令加密算法存在一个缺陷：用户的口令经上述步骤后仍保存在口令文件中，其字符代码虽已变为其他字符，但位数还与原输入口令字符数相等[4]。这为攻击者提供了方便，为了解决这一问题，可对上述口令加密算法加以改进，步骤 5 改为：将转换后得到的字符再经过 1 次变换，将 1 个字符变换为 2 个或 3 个以上字符后再保存。用 C++表述如下：

```
CString f(CString S, WORD K) // 加密函数
{  CString Kouling,zifu;
   int i,j;
   Kouling=S;
   for(i=0; i<S.GetLength(); i++)
   {  Kouling.SetAt(i, S.GetAt(i)^(K>>8));
      K = (BYTE)Kouling.GetAt(i)+K;
   }
   S=Kouling;
   Kouling.Empty();
   for(i=0; i<S.GetLength(); i++) // 对加密结果再次进行转换、加密
   {  j=(BYTE)S.GetAt(i); // 提取字符，将字符转换为两个字母保存
      zifu="12"; // 设置 zifu 长度为 2
      zifu.SetAt(0, 65+j/26);
      zifu.SetAt(1, 65+j%26);
      Kouling += zifu;  }
   return Kouling;  }
```

2.2 在第三方认证情况下的加密算法

如果系统除了管理员提供密钥和用户提供口令外，还需要第三方或第四方认证或参与才能进入系统时，可先将这些密钥进行一定的运算操作后再形成新的混合密钥。基于 C++的加密算法描述如下：

```
CString f(CString S, WORD K1,k2,k3)
// 加密函数，k1 为系统管理员密钥，k2,k3 为第三方、第四方密钥
{  CString Kouling,zifu;
   int i,j;
   Kouling=S;
   for(i=0; i<S.GetLength(); i++)
   {  Kouling.SetAt(i, S.GetAt(i)^(K1>>8));
      K = ((BYTE)Kouling.GetAt(i)+K)*k2+k3; //产生混合密钥  }
   S=Kouling;
   Kouling.Empty();
   for(i=0; i<S.GetLength(); i++)
   {  j=(BYTE)S.GetAt(i);
      zifu="12";
      zifu.SetAt(0, 65+j/26);
      zifu.SetAt(1, 65+j%26);
      Kouling += zifu;  }
   return Kouling;  }
```

3 结束语

尽管对口令文件进行加密是加强用户安全一项很好的措施，但它并不是绝对安全可靠的。其主要威胁来自两个方面：一方面，当攻击者已掌握了口令的解密密钥时，便可破译口令。另一方面，攻击者利用加密程序来破译口令[5]。如果运行加密程序的计算机速度足够快，则通常只要几个小时便可破译口令。因此，针对口令不断被破解，不断创新新的加密算法来防止攻击者的恶意攻击是我们下一步的研究方向。

参考文献：

[1] Richard C. Leinecker, Tom Archer. Visual C++6 宝典[M]. 张 艳，王文学，张 谦，等译. 北京：电子工业出版社，2001. 8.

[2] Wade Trappe, Lawrence C Washington. 密码学概论[M]. 邹红霞，许鹏文，李勇奇译. 北京：人民邮电出版社，2004. 6.

[3] 张基温. C++程序设计基础[M]. 北京：高等教育出版社，1999. 8.

[4] 汤子瀛，哲凤屏，汤小丹. 计算机操作系统[M]. 西安：西安电子科技大学出版社，2002. 9.

[5] 李闽溟，吴继刚，周学明. Visual C++ 6.0 数据库系统开发实例导航[M]. 北京：人民邮电出版社，2002. 10.

（责任编辑：黎贞崇）