管理人员甚至有些秘书都可以轻而易举得到。

在UNIX系统里有一个较好的做法。当用户登录时，登录程序首先询问登录名和口令。输入的口令被即刻“加密”，这是通过将其作为密钥对某段数据加密完成的：运行一个有效的单向函数，运行时将口令作为输入，运行结果作为输出。这一过程并不是真的加密，但人们很容易把它叫做加密。然后登录程序读入加密文件，也就是一系列ASCII代码行，每个登录用户一行，直到找出包含登录名的那一行。如果这行内（被加密后的）的口令与刚刚计算出来的输入口令匹配，就允许登录，否则就拒绝。这种方法的最大好处是任何人（甚至是超级用户）都无法查看任何用户的口令，因为口令文件并不是以未加密方式在系统中任意存放的。

然而，这种方法也可能遭到攻击。骇客可以首先像Morris和Thompson一样建立备选口令的字典并在空暇时间用已知算法加密。这一过程无论有多长都无所谓，因为它们是在进入系统前事先完成的。现在有了口令对（原始口令和经过了加密的口令）就可以展开攻击了。骇客读入口令文件（可公开获取），抽取所有加密过的口令，然后将其与口令字典里的字符串进行比较。每成功一次就获取了登录名和未加密过的口令。一个简单的shell脚本可以自动运行上述操作，这样整个过程可以在不到一秒的时间内完成。这样的脚本一次运行会产生数十个口令。

Morris和Thompson意识到存在这种攻击的可能性，引入了一种几乎使攻击毫无效果的技巧。这一技巧是将每一个口令同一个叫做“盐”（salt）的n位随机数相关联。无论何时只要口令改变，随机数就改变。随机数以未加密的方式存放在口令文件中，这样每个人都可以读。不再只保存加密过的口令，而是先将口令和随机数连接起来然后一同加密。加密后的结果存放进口令文件。如图9-19所示，一个口令文件里有5个用户：Bobbie、Tony、Laura、Mark和Deborah。每一个用户在文件里分别占一行，用逗号分解为3个条目：登录名、盐和（口令+盐）的加密结果。符号e（Dog，4238）表示将Bobbie的口令Dog同他的随机，4238通过加密函数e运算后的结果。这一加密值放在Bobbie条目的第三个域。

Bobbie, 4238, e(Dog, 4238)
Tony, 2918, e(6%%TaeFF, 2918)
Laura, 6902, e(Shakespeare, 6902)
Mark, 1694, e(XaB#Bwcz, 1694)
Deborah, 1092, e(LordByron,1092)

图9-19 通过salt的使用抵抗对已加密口令的先期运算

现在我们回顾一下骇客非法闯入计算机系统的整个过程：首先建立可能的口令字典，把它们加密，然后存放在经过排序的文件f中，这样任何加密过的口令都能够被轻易找到。假设入侵者怀疑Dog是一个可能的口令，把Dog加密后放进文件f中就不再有效了。骇客不得不加密2^n个字符串，如Dog0000、Dog0001、Dog0002等，并在文件f中输入所有知道的字符串。这种方法增加了2^n倍的f的计算量。在UNIX系统中的该方法里$n = 12$。

对附加的安全功能来说，有些UNIX的现代版使口令不可读但却提供了一个程序可以根据申请查询口令条目，这样做极大地降低了任何攻击者的速度。对口令文件采用“加盐”的方法以及使之不可读（除非间接和缓慢地读），可以抵挡大多数的外部攻击。

3. 一次性口令

很多管理员劝解他们的用户一个月换一次口令。但用户常常不把这些忠告放在心上。更换口令更极端的方式是每次登录换一次口令，即使用一次性口令。当用户使用一次性口令时，他们会拿出含有口令列表的本子。用户每一次登录都需要使用列表里的后一个口令。如果入侵者万一发现了口令，对他也没有任何好处，因为下一次登录就要使用新的口令。惟一的建议是用户必须避免丢失口令本。

实际上，使用Leslie Lamport巧妙设计的机制，就不再需要口令本了，该机制让用户在并不安全的网络上使用一次性口令安全登录（Lamport,1981）。Lamport的方法也可以让用户通过家里的PC登录到Internet服务器，即便入侵者可以看到并且复制下所有进出的消息。而且，这种方法无论在服务器和还是用户PC的文件系统中，都不需要放置任何秘密信息。这种方法有时候被称为单向散列链（one-way hash chain）。

上述方法的算法基于单向函数，即$y = f(x)$。给定x我们很容易计算出y，但是给定y却很难计算出x。输入和输入必须是相同的长度，如256位。

用户选取一个他可以记住的保密口令。该用户还要选择一个整数n，该整数确定了算法所能够生成的一次性口令的数量。如果，考虑$n = 4$，当然实际上所使用的n值要大得多。如果保密口令为s，那么通