Password Management

Matt Bishop
Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

Abstract: Issues of password management are among the most vexing in systems administration and computer security, and among the most difficult to solve. This paper surveys some of the mechanisms that have been proposed in the past, and compares their effectiveness.

1 Introduction

Probably the most common technique for user authentication is the use of passwords. The goal of this paper is to survey different techniques for associating passwords with users, and for maintaining that association. General requirements for determining if a password scheme is adequate are discussed elsewhere [1] and will not be repeated.

The first problem is that of *password selection*. Experiments conducted at Bell Laboratories [2], the Software Engineering Institute [3], and other places all confirm that a disturbingly large percentage of users' passwords can be guessed easily. Gaining access to a system is the first step to penetrating it, and so the first issue relevant to password management is the selection, or assignment, of a password that is sufficiently hard to guess so as to deter attacks.

The second problem is that of managing data associated with passwords to enable the system to determine if a user has entered the correct password. This data must be on-line somewhere, yet either must be unobtainable or else worthless should an attacker obtain it. This raises a number of interesting issues which will be explored.

We begin with a simple mathematical model of pasword systems to provide a framework for discussion. We then consider the password selection problem and the password management problem.

2 Password Systems

A *password system* is the embodiment of that information required by both the user and a computer system to authenticate the user with a degree of assurance sufficient to the computer system (and vice versa). It is composed of a set P of data called *passwords*; for our purposes we shall require elements of this set to be strings of characters (bits), but the general model does not require this. Each element $p \in P$ has associated with it *complementary data* $\overline{p} \in \overline{P}$ which is generated by a *complementation function* $c{:}P \rightarrow \overline{P}$ and is stored on the computer system. The user selects, or is assigned, a password from P using a selection function s. When the user enters a password, the system retrieves the complementary data associated with the user and applies a *password function* $f{:}P \times \overline{P} \rightarrow \{0,1\}$ to both. If the complementary data is generated by applying the complementation function to the supplied password, the password function returns 1 to indicate the user has been authenticated; otherwise, it returns 0.

Although the most common set from which passwords are drawn is the set of strings of at most some fixed length, nothe that this model encompasses other types of systems as well. For example, consider the challenge-response protocol [4][5], in which the user is sent a random string from the computer and must transform it according to an algorithm known to the user and the computer. Here, P is the set of possible algorithms, $\overline{P}$ the set of their inversions (or the same set as P, depending upon implementation), C the set containing the identity mapping(or containing the function to derive inverse transformations), S the set of algorithm choosing functions (which the user uses to select a transformation), and F the set of functions that compart the result of applying an element of $\overline{P}$ to the random string, to the result sent by the user (or that apply elements of $\overline{P}$ to the result sent by the user and compare it to the random string).

This model differs from the one in [4] by considering the non-cryptographic elements, namely password selection (the elements of the set S) and management of the password complements (the elements of the set $\overline{P}$). In this paper, we outline considerations when selecting the set S and when choosing

The support of grant NAG 2-628 from the National Aeronautics and Space Administration to Dartmouth College is gratefully acknowledged.

techniques to manage elements of $\overline{P}$.

3 Password Selection

The set S contains the functions used to select passwords. Elements of this set are implemented by programs called *password changers*. Ideally, these changers force selection of an element $p \in P$ under a uniform random distribution, so no one element is any more likely than another. Hence should a user try to guess a password (called a *trial and error attack*), on the average $|P|/2$ potential passwords must be tested.

These pasword changers are invoked either by the user or the system, and reset the user's password. The specific password associated with the user can either be assigned by the system or selected by the user. Both schemes have advantages and disadvantages.

Computer selection of passwords is recommended by the Department of Defense [6], among others. Such a scheme requires first, that the computer be able to generate a large enough set of passwords to make exhaustive search attacks infeasible; typically, this is done by producing passwords of random characters. The problem with using random passwords is that humans can repeat with perfect accuracy about 8 meaningful items (such as digits, letters, or words) [7]. If a random password were assigned for each system on which the user had an account, most users would soon be overwhelmed with the amount of random information they would need to remember.

Two sets of techniques are used to ameliorate this problem. The first is to accept the random nature of the passwords and devise a scheme to allow users to write them down. A typical example is to augment the selection of a password with a simple transformation, and then allow the user to write down the transformed password. To determine the pasword that must be entered, the user simply inverts the transformation. So, if P is the set of all strings of characters with length no greater than 8, and the transformation is to capitalize the first letter and add a hyphen at the end, the written-down password "freem3" would correspond to the actual password "frEem3-" [8]. These schema are similar to the challenge-response protocols, as well as to the use of pass algorithms [5].

The second set is to provide an alternate alphabet for the passwords. Again using the string example, each random character (or sets of characters) are mapped into one of a set of syllables [9]. This creates longer passwords, but fewer random "chunks," and hence can be remembered more easily. When used in conjunction with key crunching[10], such mechanisms can ensure that P is the same size as if it were the set of random strings of some (smaller) fixed length. But if no associations can be made, these passwords are equivalent to random passwords.

The limits on human memory are one of the reasons that allowing users to select their own passwords is so attractive; unfortunately, this technique suffers from numerous drawbacks, such as the difficulty of choosing passwords that are difficult to guess. Hence unconstrained choice is rare; usually some restrictions, requiring a mixture of case, non-alphabetic characters, and/or length are required (for example, see the *passwd* manal page in [11]). Various papers have suggested techniques for good passwords; unfortunately, their recommendations are followed all too infrequently.

Perhaps the most exciting idea in this area is the use of a very rigorous password checker. Such a program, called a *proactive password checker* [1][12], allows the system administrator to constrain the user's choice of passwords on a per-site, per-group, per-project, and per-user basis. Tests used may involve not only arbitrary lists, but also patterns, output of programs, and personnel databases. The tests are also very flexible and may be changed without recompiling the program; this allows very quick reconfiguration of the password changer.

Although not strictly related to password selection, the idea of password aging deserves some mention. Password aging is the technique of allowing passwords to be unchanged for a limited time; when that limit is reached, the user must change the password before he will be able to complete the login procedure. Its intent is to hinder an exhaustive search by requiring users to change their password before any attacker can guess it. Like computer-generated passwords, it is recommended by [6].

Simple-minded implementations of password aging are worthless, because users will simply change their password and then change it back. More sophisticated implementations require the new password to remain in effect for some time before it can be changed again (for example, see [13]); should the password be *cracked* (guessed) within this time, the user must ask the system administrator to change it.

A very serious problem with many password aging mechanisms is the lack of warning users receive before their password is to expire. Surprising users with an insistence they change their passwords during the login procedure denies then the time to think about choosing a good one; it has been observed [14] that systems with password aging often have very easy to guess passwords. So, if password aging is used, password choices should be screened very carefully, and users should be given plenty of notice before their password expires.

4 Password Management

Elements of the set $\overline{P}$ must be stored in a way accessible to the system. We assume for this discussion that the complementation functions are one-way and hence not invertible; therefore, given some $\overline{p} \in \overline{P}$, one cannot determine any password $p \in P$ for which $c(p) = \overline{p}$ except by trial and error. Trial and error attacks cannot be prevented because the attacker can simply use standard login authentication procedures; however, such repeated attempts to login will attract attention (at least on systems where security is at all a consideration). Therefore, proper management techniques should prevent attackers from performing trial and error attacks in any other way. This can be done by hiding the complementation function used, or hiding the complementary data.

The first technique requires that the attacker be forced to use the standard login procedure rather than a speeded-up version. For example, if the user has access to the complementary data and knows the complementation function, he can move the data off-line and re-implement the complementation function. Trial and error testing then becomes simple. UNIX systems are particularly susceptible to this attack, as the complementation function is well known (and has in fact been speeded up significantly; see [15] and [16]) and the complementary data is kept in a world-readable file.

The second technique simply denies the attacker the data needed to test his guesses during a trial and error attack. Kerberos [17] relies on this technique, in that it storeds the complementary data in a physically secure authentication server not accessible to the user community except through the authentication protocols. So-called shadow password files in UNIX systems (see [3]), which put the complementary data associated with each user in an unreadable file, also use this approach.

In challenge-response protocols, the password management problem becomes one of managing algorithms rather than strings. Provided the algorithms are chosen from a rich enough set, this can be made very difficult by storing the algorithm only in an executing program and protecting memory. It is (usually) much easier to steal a file from a computer system than to steal a copy of the memory image of a process.

5 Conclusion

This paper has very briefly discussed two of the most problematic, and yet vital, of the parts of password systems. Using a simple yet powerful model, we have described ways to select passwords and identified two techniques to hinder the compromise of a system by guarding the information and algorithms used to validate user passwords.

Obtaining access to a system, or to resources on the system, is the first step in attacking the system. Penetration by obtaining, or guessing, a password is a time-honored, and extremely effective, technique for gaining such access; so a firm understanding of passwords, their uses, and techniques for password management are essential to the security of any computer system.

References

[1] M. Bishop, "Password Checking Techniques," *Proceedings of the Workshop on Computer Incident Handling* (June 1990) pp. III-D-1:4.

[2] R. Morris and K. Thompson, "Password Security: A Case History," *Communications of the ACM* 22(11) (Nov. 1979) pp. 594-597.

[3] D. Klein, "Foiling the Cracker: A Survey of, and Improvements to, Password Security," *Proceedings of the UNIX Security Workshop II* (Aug. 1990) pp. 5-14.

[4] G. Brassard, *Modern Cryptography: A Tutorial*, Springer-Verlag, New York City, NY (1988).

[5] J. Haskett, "Pass-Algorithms: A User Validation Scheme Based on Knowledge of Secret Algorithms," *Communications of the ACM* 27(8) (Aug. 1984) pp. 777-784.

[6] *Department of Defense Password Management Guide*, CSC-STD-002-85 (Apr. 12, 1990).

[7] G. Miller, "The Magical Number Seven Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *Psychological Review* 63 (1956) pp. 81-97, cited in [18].

[8] M. Crabbe, "Password Security in a Large Distributed Environment," *Proceedings of the UNIX Security Workshop II* (Aug. 1990) pp. 17-30.

[9] M. Gasser, "A Random Word Generator for Pronounceable Passwords," ESD-TR-75-97, Electronic Systems Division, Hanscom Air Force Base, Bedford, MA (Nov. 1975).

[10] L. Grant, "DES Key Crunching for Safer Cipher Keys," *SIG Security Audit and Control Review* 5(3) (Summer 1987) pp. 9-16.

[11] *UNIX Users' Reference Manual, 4.3 Berkeley Software Distribution, Virtual VAX-11 Version*, Computer Systems Research Group, Computer Science Divicion, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720 (Apr. 1986).

[12] M. Bishop, "A Proactive Password Checker," PCS-TR90-152, Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH 03755 (June 1990).

[13] P. Wood and S. Kochan, *UNIX System Security*, Hayden Books, Indianapolis, IN 46268 (1985).

[14] F. Grampp and R. Morris, "UNIX Operating System Security," *AT&T Bell Laboratories Technical Journal* 63(8) part 2 (Oct. 1984) pp. 1649-1672.

[15] M. Bishop, "An Application of a Fast Data Encryption Standard Implementation," *Computing Systems* 1(3) (Summer 1988) pp. 221-254.

[16] D. Feldmeier, "A High-Speed Crypt Implementation," to appear in the *Proceedings of Crypto '90*.

[17] J. Steiner, C. Neuman, and J. Schiller, "Kerberos: An Authentication System for Open System Networks," *USENIX Conference Proceedings* (Winter 1988), pp. 191-202.

[18] C. Coombs, R. Dawes, and A. Tversky, *Mathematical Psychology: An Elementary Introduction*, Mathesis Press, Ann Arbor, MI (1981).