Global Information Assurance Certification Paper

One-time Passwords

Gordon Fullerton

19 September, 2000

There has been a significant amount of discussion and reams of documentation on the proper length, structure, and format of user ids and password. Debates abound on what is the proper period or time frame between password changes. In reality, the best time frame between password changes is every time they are used.

Many systems are structure to the extent that they maintain records for many previously used (sometimes as many as 10) passwords. This is in the belief that reusing a password used a year or more ago is a security risk. Fore example, if a required change period is 90 days and the system will not allow you to reuse a password for 5 change periods, you will not be able to reuse a password for over a year and a half. Systems blocking reuse for 10 change periods result in prohibiting reuse for over 3 years. Somehow, there is a security risk I a user uses the same password in 899 days. But, what about the user using the same password for 89 consecutive days. That consecutive use would seem to be more of a potential risk than reusing a password after two and one half years.

The risk is simple to see. Regardless of how well structured a password is, no matter how long or complicated it is, even though you mix alpha with numbers with special characters; it can be discovered.

If you give a user a complicated password, or force them to select a complicated one, the user will typically write it down. How many times have we found password written on the wall, taped to the bottom of the keyboard, or even stuck on the monitor with a little 'yellow sticky'. Even, worse, if the password is used by the legitimate owner, it can be 'sniffed' off the network. There are a number of powerful, easy to use, and free sniffers on the Internet. The potential hacker simply installs the software and 'listens' for user Ids and associated passwords to flow through the network. How simpler do we need to make it for anyone to uncover the system admin user id and password for the server(s) in the network.

Another common method of compromising a system that is protected by well managed and implemented password systems is to compromise the passwords stored on a system. Systems that utilize common methods of password control must store the correct password on the system in order to compare it against that password presented by the user. Hackers can get passwords that are stored on the system. There are many exploits available on the Internet to enable the cracking of password files. Once cracked, the passwords stored in that file are vulnerable to misuse.

In their identification of methods to improve system security, the CERT at Carnegie Mellon recommends the use of one-time passwords as a method of securing password

files on servers [1]. There are different approaches available to implement one-time passwords.

S/Key is an example of public software that can be used in a one-time password environment. Originally developed at Bellcore, it is available at their ftp site or a successor OPIE is available from the NRL FTP site. A Java applet that calculates an OTP is available at http:\\www.cs.umd.edu/~harry/jtop/ [2].

The S/Key process provides protection against sniffing a password [3] from the network but still requires a protected password file to exist on the host server. To use S/Key or one of it's derivatives, the user must select a secret password (like the pass phrase used in PGP) to feed the secure hash generator. The generator applies the hashing function n times against the secret password where user has selected n. The output is saved on the server to use in future access attempts. For the user to access the system they must have either a software or hardware S/key generator. To log on the user identifies themselves and is given a prompt which is simply n-1 (starting with the original user selected n but where n is decremented every time it is used). Knowing the secret password, the user applies n-1 iterations of the secure hashing function to generate the response to the server. The serve checks that against the n-1 stored answer and if correct allows access. In this manner, each access results in a new or different 'code' transmitted over the network. If a hacker captures the transmission, the hacker only has an old, no longer unusable password that is of no value.

However, as noted, this process is good protection from sniffing but fails in other risk areas. It does not protect against the user writing the secret password on the wall for any one with and S/Key function generator to compromise. It also does not protect against a hacker being able to access the stored password file on the server and using the contents to gain access to high level privileges or other systems.

Fortunately, there are other one-time products available which address those issues. One method is to setup a time based synchronization involving a central authorization point and individuals. Typical of this are the SecurID products from Security Dynamics. This system uses hardware tokens similar to a credit card, which has and LED and onboard microchip. Each card is synchronized to a central point and provides, via the LED, a new password every 60 seconds. To access a system, the user logins in with user name and at the prompt looks at the token for the password then enters the password currently shown on the token. The system being accessed takes that information and quires a central authorization server to see if the password is correct for that user for that time period. The authorization server then sends back either an OK or denies service response to the initial system. Key to this approach is that the authorization serve does not allow login from remote or network users and is dedicated to only serving as an authorization server. If not then there is the potential for a hacker to gain access to the server and compromise the authorization process.

Even with this approach to protecting passwords there are careful administrative steps

that must be followed. While the token/password can not be pinned up on the bulletin board, the user must maintain its integrity. Also, default parameters on system protected are critical. For instance the system administrator must be concerned about what happens when the authorization server can not be contact. Unless modified, some systems will default to open access. Those defaulting to a stored password list must have a list that only uses strong passwords; not guest, system, or null.

There are many commercial off-the-shelf systems that include one-time password functionality. Often vendors team with another to incorporate some of the more popular software implementations into their own hardware. Two commonly used remote access systems are Shiva and Citrix. Shiva supports one-time passwords generated by products from Digital Pathways and Security Dynamics [(4)]. Citrix supports the Digital Pathways product Defender 5000[(5).]

Using commercial products or providing support to public systems does cost time and money. Faced with the common trade off of, security verses usability verses costs, a common practice is to implement one-time passwords only for critical or high-risk accounts. By protecting user ids and passwords of those accounts with high level privileges only the general user accounts are exposed to hacking or sniffing. If compromised, the hacker has a foot hold for further exploits but not system admin or root access.

(1) Carnegie Mellon Software Engineering Institute, "Protecting Yourself from Password File Attacks" URL: HTTP://www.cert.org/tech_tips/passwd_file_protection.html. (19 September, 2000)

(2) Harry Mantakos, "jtop: The Java OTP Calculator", URL: http://www.cs.umd.edu/users/harry/jotp/. (18 September, 2000)

(3) "The S/Key Password System", URL: http://www.ece.nwu.edu/CSEL/skey/skey_eecs.html (14 September 2000).

(4) "Digital Pathways", URL: HTTP://comet.shiva.com/prod/docs/nilibrary/bi0415.html, (19 September, 2000).

(5) "Digital Pathways Defender 5000", http://www.citrix.com/library/SGs/wfsolg/www/SolgDigital_pathways_Defender_5000.htm (18 September, 2000).