Global Information Assurance Certification Paper

Password [In]security:

Common issues surrounding compromised passwords

Carl Root, Security Essentials Practical, Version 1.2d

In the world of network security, one of the first and simplest lines of defense that is usually implemented against unauthorized access to a network, local computer, or software application is password protection. Unfortunately, as we have learned through many a cyber incident, passwords, in some cases, may be of little or no defense at all. In the following pages, we will set out to take a look at some of the problems surrounding password protection, and then discuss some of the industry solutions, such as Secure Single Sign On, which are currently being implemented in hopes toward improving the problem surrounding compromised passwords.

One of the first issues we should address, perhaps even the most key issue concerning password security, is the fact that user intervention is one of the elements. Many corporations devise password policies, which describe the criteria that users must follow in creating adequate passwords. Unfortunately, whether for convenience sake or some other reason, most users do not follow the password policies set forth by their organization. Instead, they choose passwords that are easy to guess, or crack, resulting in a potential security breach. O'reilly's book, Computer Security Basics, further illustrates this point, [1] "Studies indicate that a very large percentage of users' passwords can easily be guessed. With the help of online dictionaries of common passwords (English words, names of people, animals, cars, fictional characters, places, and so on), crackers are quite likely to be able to guess a good many of the passwords most people are likely to choose." Corporations may spend hundreds, even thousands of dollars in training and awareness, but in the end, it is still the users' choice.

There are, though, ways of helping to subdue this. Network Administrators may be able to set policies on users' accounts, specifying certain criteria that the users must follow in selecting a password. Some common guidelines may specify that a password has to contain an upper and a lower case letter, one number, or one special character, such as a period or exclamation point, must be a specific length, or any combination of the above. If your operating system or application does not support this, however, there are third party utilities available as well that may allow you to add that functionality. The end result is if the user does not choose an acceptable password, the operating system or application will not accept the password that he or she has chosen, and they will then be prompted to choose one that follows the set policy.

Another commonly followed practice is to set expiration dates, forcing users to change their passwords periodically. Although this may not prevent a user from picking an easy password, the idea behind this is if a users' account was to be compromised for any reason, the hacker / cracker would not have unlimited time to use the password he obtained to gain unauthorized access to your network. Although setting password expiration is better than not setting it at all, it still does leave a window of opportunity. At the complaint of users who must change their password frequently, most Administrators

set the expiration at thirty, sixty, or sometimes even ninety days. The longer the duration, the more chance a hacker has to steal a password, and use it to infiltrate your network.

In conjunction with setting password expiration, an Administrator may decide to set password uniqueness restrictions, also commonly known as password history. A relative article from the November 1994 edition of Windows NT for Professionals explains, [2] "The Password Uniqueness restriction works with the Maximum Password Age restriction, which specifies how often your users must change their passwords. The Password Uniqueness restriction is designed to prevent the users on your network from recycling a password." This would increase security in the fact that if a password was to be stolen, the user cannot use his current password again until after it has been changed a set amount of times, should he be tempted to do so. For example the administrator may set the account to remember up to ten passwords. The user would then not be able to use his same password again until the eleventh password change occurs. Therefore, the stolen password also cannot be used once the current password expires.

In addition to the account policies set above, an Administrator also may set account lock out restrictions, which locks out a users account after a number of unsuccessful login attempts are made. This usually deals with three areas; how many attempts allowed before locking the account, how long to lock it out for, and how long to reset the count between each pair of failed login attempts. These may change from organization to organization depending upon password policy. One example taken from an online article by Luqman Mahmud, Data Security Administrator for Federal Express states, [3] " The account should be locked out after five attempts, the count should be reset after thirty minutes, and the duration of lockout should be indefinite." This is exceptionally good, since it offers an extra level of security by requiring administrative intervention in order to re-instate the users access rather then having the account lockout automatically cleared after a pre-determined amount of time.

Setting account restrictions may help improve security in these areas, but there are other areas to consider concerning password security as well. In an effort to help them remember their passwords, users may also be tempted to write them down and leave them in a not so inconspicuous, or inconspicuous but accessible location allowing virtually anyone who walks by free access to their workstation. Users also may, "in good faith", share passwords with fellow employees, such as the support person who needs to log into their machine repeatedly, or the guy visiting from the sales office in Seattle who just wants to check his e-mail. Michael E. Kabay, PHD, further illustrates, [4] "Sharing a password with someone for momentary convenience compromises security. The lender might forget to change the password, providing an open door until the next change. Users who choose passwords poorly can reveal patterns for preferences that make it easier to guess the next password. For example, if a poorly chosen password is feb02mypass in February, what do you think the password for March might be?" Users should be strongly cautioned, and educated against such things by their organization. Not only does this pose a security threat, but any logged activity related to that specific account can no longer be attributed to that user.

Unfortunately, there is no operating system level, or application level policy enforcement that will help solve the problems of written down or shared passwords. There are though, methods currently being implemented in the industry that may very well be a step in the right direction toward solving some of the problems we have been discussing concerning password issues.

One possible solution is the one-time password method of authentication This method exists in many different forms in the industry and usually requires a third party utility to be installed, or in some cases, even adding to or reworking your existing infrastructure. In the following few paragraphs, we will begin to show some examples of one-time password authentication methods and briefly explain how they work.

Token authentication is one of the ways in which we see one-time password authentication surfacing in the industry. In his June 1997 article from Windows 2000 magazine, Ben Rothke makes an excellent claim as to the increased security of using tokens as a secure password alternative, [5] "Token-based authentication eliminates nearly all the risk involved with validating users in a network. Token-based schemes improve security, lower per-user cost, centralize and reduce administration costs, and minimize unauthorized access to services."

There are a few different types of tokens, one of which displays a random number every sixty seconds. That number, along with a pin number created by the user during the initial logon process is his password. The idea is that since the user is using a different password each time, it then would be much more difficult to compromise the account. Since the number changes every sixty seconds, by the time a hacker or cracker cracked your password, it would essentially be no good. Another type is similar to a keypad, which resembles a small calculator. Upon logging onto the system, a number is displayed for you to enter into the token. After entering the displayed number into the token, the token will then display a different number back to you for you to use as your "password" to enter into the system. The token itself is also secured by a pin number, which you choose when you receive the token, which will unlock it for use. This is in case the token inadvertently falls into the wrong hands.

Another method we sometimes see deployed in the industry is the use of actual one-time passwords. As the method described above, this also requires some type of third party software to be involved, usually a client interface and its counterpart, a server-side agent. The idea is that the software can be used to generate a list of passwords that can only be used once for authentication and then discarded. Since the software generates the password list, the client and server agent are coordinated as to which passwords are going to be used. The user need only pick one password from the list, use it one time, and then discard it. Upon entering the password, it is encrypted by a one-way hash function before it is passed to the server agent. This is done for two reasons. One, so that the password may not be sniffed during data transmission, and two, so that if a hacker were to try and crack the stored password, he would need to reverse the hash function, or supply the exact same hash in order to be authenticated instead of the actual user. In some instances, vendors may design their software to pass the users password through the hash function

more than once incrementally, making it virtually impossible for even the most elite hacker to supply the same hash, or crack the existing one.

But password sign on is not the only means of authentication available to us today. Much research and technological advancement have brought us the capability of using such things as biometrics as an alternate, or additional means of authentication. Biometrics is the use of devices that allow such things as voice, face, or fingerprint recognition in order to gain access to your workstation. In the example of fingerprint recognition, you would simply place your finger upon the designated biometric finger pad. The device would then read your fingerprint and match it against a vector template of your stored fingerprint pattern. These vector templates are recorded from your actual fingerprints, which you must supply upon initial system setup. The actual fingerprint itself is not stored, because of privacy concerns. An actual fingerprint cannot be re-constructed from the stored template. When a match is found, access is granted. In some cases, for extra authentication, a password may have to be entered or a smart card used in addition to entering your fingerprint. Either way, we can see that this is much more secure than conventional password authentication.

There are different ways of combining these methods as well in an effort to take authentication even one step further. Secure Single Sign-on is gaining acceptance in the industry as a more secure authentication alternative. In this method, a user would simply sign on once using one or a combination of the methods discussed thus far and, without intervention from them, would be signed on to every application and / or user account that they possess. There are many advantages to this type of authentication. One advantage is that the user is less involved in the authentication process, so the "passwords" which are ultimately passed back and forth automatically while signing on can be much more secure than any user selected password. A second advantage is that it would minimize the time, money, and resources currently involved in daily (or in some cases hourly) password resets. Also, it is much easier from a user standpoint to enter and remember just one password and supply a fingerprint than to remember five to ten passwords depending upon how many accounts they may have. Tim Tervo in his technical whitepaper, Single Sign-on Solutions in a Mixed Computing Environment, further explains some of the advantages, [6] "Single Sign-On systems relieve administrative burden from both the users and administrators. The users do not have to manage many passwords by themselves, but they can use only one to log on to all systems. This will allow them to concentrate more on the work at hand instead of worrying about the forgotten passwords after a vacation. The passwords with all of their deficiencies can be eliminated altogether with use of stronger authentication methods, like hardware tokens or digital certificates. This significantly adds security to the system." So we see, Single sign-on not only adds increased security, but ease of use for the user as well.

There are, though, some disadvantages to the Single Sign-on model. Problems concerning lack of support from all operating systems as well as problems surrounding the revocation of user access in a heterogeneous environment are still among the issues, which need to be resolved. Single Sign-on authentication can be achieved by different

means, some of which can be costly, especially where Biometrics are concerned. Should an organization decide that they want to use Biometrics for Single Sign-On authentication, they may have to invest thousands of dollars in hardware alone depending upon the size of their organization. Single sign-on also may require an entirely separate infrastructure in order to implement it. This can involve months of planning. One statistic showed an average of twelve to fifteen months to deploy most sign-on solutions. There is also the possibility of needing to hire additional personnel depending upon the scope of the project. Money may have to be spent after the fact on organization wide user training. In the light of the cost of recovering from a security breach or repeated security breaches, which could result in stolen proprietary information and damage, it may be worth the expense. There are, of course lower cost alternatives such as scripted workstation sign-on, which is simple and cost effective to deploy. The major disadvantage to this simple alternative is that if a hacker were to somehow compromise that single user password, the results could be devastating. It would be rather difficult for a hacker to supply your fingerprint.

Every time a password is stolen, cracked, shared, or written down it is a constant reminder of the need to find a more secure solution, preferably one that involves less user intervention. Setting account policy parameters to discourage users from choosing easy passwords, setting passwords to expire, and other related administrative attempts do help some, but not enough. Secure Single Sign-on, as well as some of the other methods we discussed, may not be the end of the issue, but they do offer more secure alternatives than some of the previously available methods. So much so that many large organizations are calculating the cost and jumping on the Single Sign-on bandwagon. This may help in seeing a future of more integrated Single Sign-on capabilities and an emergence of better Sign-on models.

In conclusion, we have discussed some hopeful possibilities of improving password security. Unfortunately though, this is just one small aspect of information security as a whole. There are hosts of other possible ways that a hacker may attempt illegal entry into your network. New exploitable vulnerabilities seem to surface every day. Viruses can be propagated nearly worldwide with just one click of the "send" button. It appears, though that the biggest concern is not an external one. [7] "Although the threats from external attacks are real, they are not the principle source of risk. FBI statistics show that more than 60% of computer crimes originate inside the enterprise." This is a staggering thought. We can invest time, money, and resources into improving password and other areas of security, but who would have ever thought one of the biggest problems would be trying to protect your information against the people who are supposed to have access to it in the first place.

References

[1] Russell, Deborah, Gangemi, G.T., Computer Security Basics, O'reilly & Associates, Inc., USA, 1991 p.60

[2] Anonymous, Understanding the Password Uniqueness Restriction, Exploring Windows NT for Professionals, November 1994
http://www.elementkjournals.com/ewn/9411/ewn4005b.htm

[3] Mahmud, Luqman, Procedures for Hardening Windows NT Workstation, Data Security Administrator, Federal Express Corporation
http://www.user.fast.net/~lmahmud/index4.html

[4] Kabay, Michael E., PhD, The NCSA Guide to Enterprise Security, Protecting Information Assets, McGraw-Hill & Companies, Inc., NY, USA, 1996 p.158

[5] Rothke, Ben, Token-based Security Add-ons, Windows 2000 Magazine, June 1997
http://www.win2000mag.com/Articles/Index.cfm?ArticleID=526

[6] Tervo, Tim, Single Sign-on Solutions in a Mixed Computing Environment, Helsinki university of Technology, Nov. 1998
http://www.hut.fi/~totervo/netsec98/sso.html

[7] Anonymous, Comprehensive Enterprise Network Security Assessment: A Whitepaper for Enterprises in an Internet Environment, date unknown
http://secinf.net/info/ids/censa/CNSWP.html