GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

Solving the Password Problem

Phillip Runner
GSEC Practical 1.4b, Option 1
Submitted July, 2004

1.0 Abstract

The protection of an organization's critical information is often only as good as the weakest link in an organization's security infrastructure. An organization may have many good security measures put in place (e.g. packet inspecting firewalls, de-militarized zones for internet-facing systems, intrusion detection/prevention systems, etc.) but if an attacker can go around those security measures and break into a poorly protected user account, their job can be made that much easier. Unfortunately, for many organizations a password is the only defense in place between an attacker and the information he/she is seeking. This paper will examine the use of passwords for authentication, the technical and social weaknesses of passwords, and then look at how biometrics and security tokens can improve the security of the authentication process.

2.0 A word about Identity, Authentication, and Authorization

Before we can talk explicitly about passwords, we must understand the concepts of identity, authentication, and authorization. Without a proper understanding of these terms, we cannot really understand the purpose for having a password. Identity is generally understood to be what something is or who someone is. Both a computer and a human have a specific identity. A human may be identified by his name, or physical characteristics. A computer may be defined by a specific hardware configuration or unique network Media Access Control (MAC) address. In both cases, however, they have a unique identity. Authentication is defined as "the process of determining whether someone or something is, in fact, who or what it is declared to be"[1]. While there is a similarity here with identity, the authentication process seeks to prove that something or someone is who or what they claim to be. Authorization allows something or someone to actually do something. A user may be authenticated to the system (they are who they say they are), but may not have the authorization to do anything.

Let's look at the example of a credit card transaction. John Smith decides he wants to buy a $2900 engagement ring for his girlfriend. After weeks of searching for the perfect ring, he finally finds it. Amazingly it doesn't need to be re-sized, so he can take it with him from the store. The clerk packages it up and asks how John wants to pay. He pulls out his wallet and digs for his credit card. The clerk runs the card through the reader, prints out a receipt and asks John to sign it. He also asks to see a copy of John's drivers license. After John signs it, the clerk examines the signature to make sure it matches the signature on the back of the card. He also examines the driver's license to verify that John matches the picture. John's identity is being authenticated to prove that the credit card is actually his. Now, for example, let's say John's credit limit is only

[1] Authentication, searchSecurity.com

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

$2000. The clerk runs the card through the reader and is returned with an error: Insufficient credit. John is not authorized to charge $2900 to his card! With the proper authorization, the transaction would be processed. All three factors are important in determining proper access control. If any part fails, access to the desired resources is denied.

In the world of information security, passwords are often used as the sole measure of authentication. A user provides a username (his identity), and a password to authenticate that they are who they claim to be. If a valid combination is given, access is granted to the resources the user is authorized to use. Because passwords are often the main line of defense, we must make sure they are hard to guess and hard to break. The following section will examine some of the problems that exist with using passwords as the primary means of authentication.

3.0 Problems with Passwords
3.1 Technical Problems

3.1.1 Algebraic Problems

A major problem with passwords is that they are simply a string of characters. Because of this attribute, a password can only have so many possibilities and is subject to brute-force guessing. An attacker could continuously try to logon to a user's account (given they have obtained a valid userID) with all combinations of passwords until they find the correct one. Thankfully, vendors have provided solutions to make this unfeasible to attackers. Most network operating systems will temporarily lock an account after a set number of incorrect login attempts over a period of time. This is called intruder lockout. After a certain timeout period, the lock is automatically lifted. An automated password guessing attack would be forced to take a lengthy period of time, yet because the lock is automatically lifted, security administration and/or helpdesk staff will not have to manually unlock the account.

In strictly mathematical terms, the strength of a password can be determined by the "password space," or the possible number of passwords that can be used. As we will see later, the password space can be greatly reduced by some of the social flaws associated with passwords. Mathematically, however, the password space can be determined by the formula:

$$P = C^n$$

where P is the total number of possible passwords, C is the total number of unique characters that each place in the password can hold, and n is the length of the password.[2] For example, if a four character password consisted of only letters and numbers, $P = 36^4 = 1{,}679{,}616$ possibilities. With the high speed of today's computing technology, a brute force attack of this nature would quickly break a password of this strength. The strength of passwords, then, increases exponentially with the length and

[2] Beverstock, 4

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

size of the character set used. For a complete analysis of the algebraic issues associates with passwords, please see the article "Passwords are DEAD! (Long live passwords?)" by David Beverstock.

3.1.2 Password Cracking

One way attackers commonly try to harvest user passwords is through a technique called password cracking. SearchSecurity.com defines a password cracker as "an application program that is used to identify an unknown or forgotten password to a computer or network resources."[3] If an attacker can gain access to a password file, they can try to recover the passwords for all the accounts in the file. They can do this offline, rather than actually attempting to authenticate to the system numerous times (and possibly triggering IDS sensors). Password crackers work by encrypting a list of passwords using the same algorithm that the passwords in the file are encrypted with. They then compare the encrypted guesses with the entries in the password file and can deduce the plaintext password for any matches!

There are three main types of attacks done by password crackers: the dictionary, hybrid, and brute force attacks. The dictionary attack is the simplest, where the cracking program compares a list of dictionary words against the password file. Because many users choose simple passwords, this method will typically reveal many passwords. The hybrid attack is a combination of the dictionary attack with additional variations. For example various numbers may be added to the end of dictionary words to create additional passwords to check against. Many users will simply add a number to a normal word for their password, so a hybrid attack can be very successful. Finally, the brute force attack checks all possible combinations of characters against the file.[4] A brute force attack will always find the plaintext password; it is just a matter of how much time it takes. The more complex and lengthy a password is, the longer it will take to crack the password. Because brute force attacks will always find the password, it is good practice to require that passwords be changed after a certain interval of time. This interval should be determined based on the enforced complexity of the password and how long, on average, it would take to crack a password of that strength.

There are many programs that can be used to crack passwords. Some of the more common ones are LC5 (formerly L0phtcrack) and John the Ripper. LC5 is primarily used for Windows password cracking, however this newest version now provides support for Unix[5]. John the Ripper is primarily for Unix password analysis, however, it can also crack Windows passwords once the password file is extracted and converted into the format of a standard Unix password file.

[3] password cracker, searchSecurity
[4] Shimonski
[5] LC5 Datasheet

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Windows Password Cracking

Although all passwords are crackable, Microsoft has made it significantly easier to crack many of theirs. Before Windows NT 4 SP4, Windows supported two kinds of challenge/response authentication: LanManager (LM) and Windows NT Challenge/Response (NTLM). To be backwards compatible, however, before SP4, Windows NT clients always used both LM and NTLM schemes to authenticate, even if the server supported NTLM. Unfortunately, LM is significantly more insecure than NTLM for several reasons. First, LM breaks the password down into two seven character chunks and encrypts them separately. To break the password, one must break two smaller passwords, which is much easier to do than breaking one long password. In addition, LM converts all lowercase letters to uppercase, greatly reducing the password space the user must attack.[6]

One of the best ways to mitigate these types of password attacks is to disable LM and NTLM authentication and migrate to NTLMv2 authentication if possible. NTLMv2 is available for Windows NT domains with SP4 or greater, and is native to Windows 2000, XP, and 2003 domains. NTLMv2 can also be installed on Windows 9x clients if the Directory Services Client is installed on those machines. For complete details, please see Microsoft knowledge base articles Q239869[7] and Q147706[8].

LC5, the newest version of L0phtCrack, is an excellent tool for cracking Windows passwords. It has a very nice wizard that will easily guide you through getting passwords to crack (from the local machine, remote machines, an NT repair disk, or through network sniffing), how to crack them (dictionary, hybrid, or brute force) and how to display the results when done. Some features included in LC5 are as follows:[9]

- Automated and Schedulable Password Scanning
- Windows *and* Unix support
- Remote System Scans – multiple domains or systems
- Uses pre-computed password tables containing trillions of entries
- Multiple dictionaries, including ones with international characters
- Password quality scoring – gives a score on how strong recovered passwords are.

In a simple test of LC5, I ran a hybrid attack against nine users with passwords of varying strength. The test was run on an Intel 800 Mhz Pentium III processor with 256 MB of RAM. When using Hybrid mode, LC5 first runs the simple dictionary attack and then follows up on anything it did not crack with the hybrid attack. In my test, two accounts were cracked within 2 seconds using the dictionary attack. In addition, a third account was discovered via the hybrid attack after 33 seconds. Two other accounts were partially broken – the second halves of the LM hash were cracked. Running only

[6] 'How to Disable LM Authentication on Windows NT'
[7] 'How to enable NTLM 2 authentication'
[8] 'How to Disable LM Authentication on Windows NT'
[9] 'LC5 Feature Comparison Chart'

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

a total of 46 seconds, 3 of 9 accounts were completely cracked and parts of 2 others were cracked!

Figure 1 shows some of the reporting and charting that LC5 can give regarding passwords it cracked. By default it creates graphs/reports on:

- Risk of the passwords cracked (high, medium, low, empty)
- Password character set (alphabetic, alphanumeric, etc.)
- How the password was cracked (dictionary, hybrid, brute force)
- Length of passwords cracked

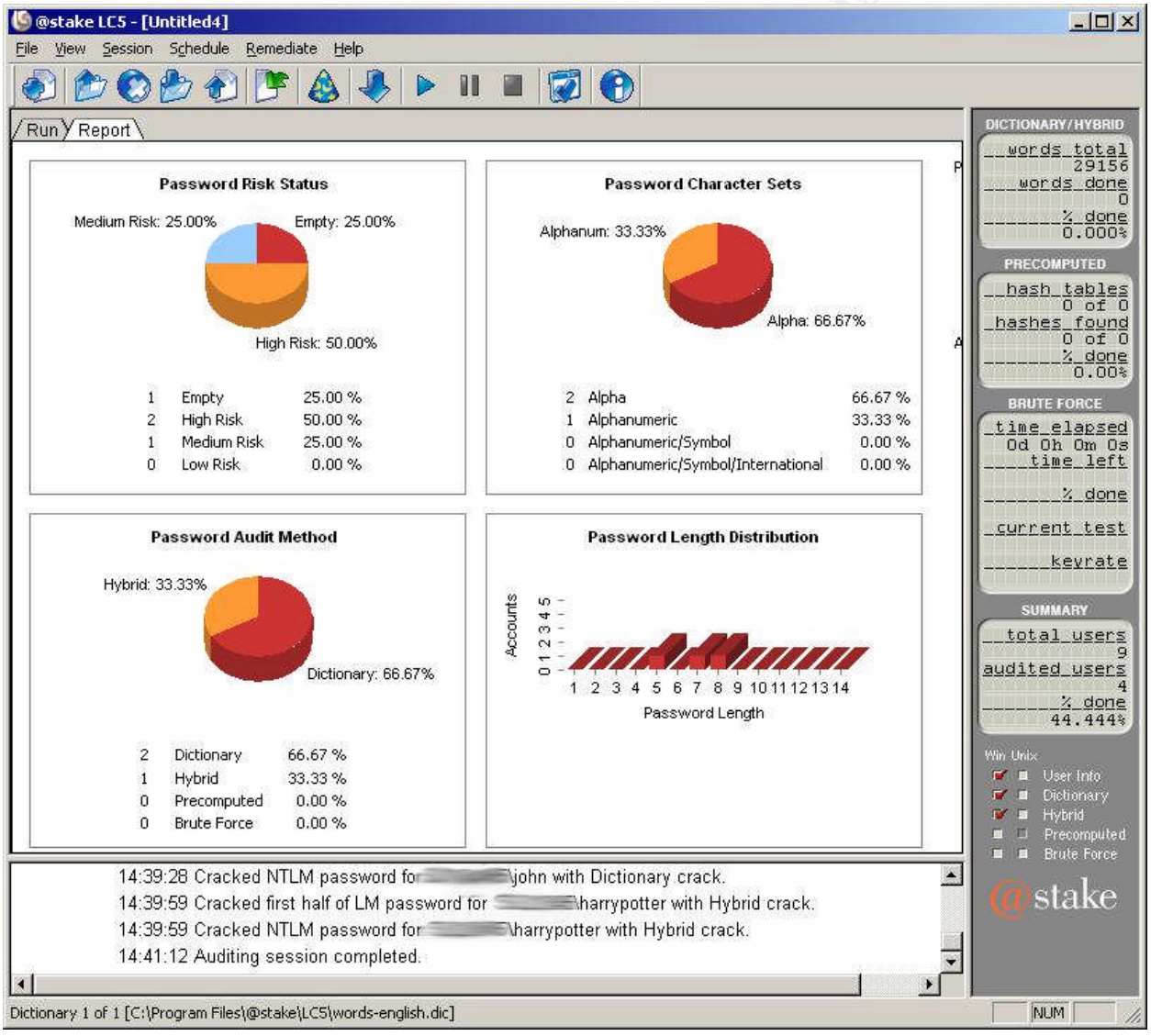

Figure 1

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

As we have seen, passwords are very susceptible to cracking attacks. To mitigate these risks passwords, if used, must be hard to crack. Passwords must not be any dictionary words, or words with numbers or characters tacked on the end as password crackers can crack these relatively easily. They also should not be words spelled backwards as brute force engines can easily break these. A good password should be strong in three main areas: length, width, and depth.[10] Typically the longer the password, the stronger it is as there are more combinations a cracking program must check. The width of the password involves the types of characters that are in the password. A password is stronger if it uses lowercase, uppercase, numbers, and special characters because the password space is that much bigger. Finally, password depth addresses password meaning – how to choose a password that is easy to remember, but hard to guess. Humans have a much easier time remembering phrases than a random string of characters. Therefore, it can be helpful to make up a password using the first letters of all of the words in a sentence, substituting some letters with characters. For example, the phrase "The dog chewed my wallet, destroying my cash" could become the password "tdCmwDm$'. This password would be very hard to crack using a traditional password cracker.

As we will see in the next section, though, just because an organization requires strong passwords, does not mean that the password problem is solved.

3.2 Social and Behavioral Problems

In April 2004, a survey on passwords was done for the Infosecurity Europe trade show in London. The results were astonishing! The survey found that over 70% of people would exchange their computer password for a chocolate bar! In addition, 34% of those surveyed would give away their password without any stimulus! Of those remaining, 34% would reveal their password when questioned if it had anything to do with a pet or child's name![11] Now these numbers may be a bit high – many may have lied and gave a false password to get the chocolate – however, the fact that people seem to have such little concern for protecting their passwords is shocking.

This example shows one of the prominent problems associated with password authentication: the human factor. Gartner analyst Rich Mogull believes that social engineering (the manipulation of people rather than technology to bypass security measures) is the greatest security risk in the next decade and that most damaging penetrations will be due to social engineering, not hacking.[12] Because humans by nature are vulnerable to manipulation, password information can often be easily stolen from an end user. The example about trading a password for a chocolate bar shows this quite clearly. Humans also typically want to be helpful. This can also play into a social engineer's hands. Take, for example, a typical office employee. A social engineer may pose as someone from the Information Technology department, call the user, and claim to be working on a project to upgrade some software on the user's

[10] Granger
[11] 'Passwords Revealed by Sweet Deal'
[12] Mogull

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

workstation. However, to do this they must have the user's ID and password to install the software. Many users, wanting to be helpful (and get the new software), will willingly trade the information. Using social means, the attacker has just harvested a userID and password that can be used to logon and either gather more information or begin their attack!

Several characteristics of passwords themselves make them especially subject to social attack. First, because passwords are simply a character string, they are something that must be remembered. Because of this principle, humans will naturally want to pick something simple. They will often pick a single word, the word 'password', the name of the organization they work for, the name of a family member or pet, or something else they can easily remember. With a little bit of research, a social engineer can harvest many of these passwords simply by guessing! If an attacker can obtain the password file, a cracking program will easily break any standard dictionary words that are used.

Because of these problems, security administrators often force a password to meet certain strength requirements. For example, a password might be required to have at least 8 characters and be alphanumeric. Unfortunately, restrictions like this lead users to write passwords down on sticky notes for all to see, use the same password on every system (giving easy entry to an attacker that discovers one of the user's passwords), or save password information in an unencrypted file on the local hard disk. When administrators use password histories to require users to choose different passwords each time they expire, users will often use the same password over and over, incrementing just one number within the password. For example, the first time a user's password may be 'winter1'. The next two times that user changes their password, it becomes 'winter2' and 'winter3.' If an attacker learns this pattern, it will be easier for them to harvest the account.

As we have seen, passwords have many problems associated with them. Even if the technical cracking techniques can be mitigated through the requirement of strong passwords, we see that attackers can still easily gain access through social means. The next section will explore some alternatives that can be used for authentication.

4.0 Alternatives to Passwords

There are four ways that users are typically authenticated:[13]

- What you know – passwords or passphrases
- What you are – static biometrics – physical characteristics like fingerprints or retinal scans, etc.
- What you do – dynamic biometrics – speech recognition, signature, etc.
- What you have – tokens, smartcards, photoIDs, etc.

[13] Kabay, 6/1/04

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

We have already seen that passwords have many problems and are inadequate for protecting our critical systems. We will next explore some of the advantages and disadvantages that biometrics and tokens provide.

4.1 Biometrics – What You Are

There are many different types of biometrics available for user authentication. Fingerprints, retinal scans, DNA tests, face recognition, iris scans, and voice recognition are just a few of the biometrics available on today's market. Before we look specifically at how fingerprints can be used for authentication, an overview of how biometrics in general are used for authentication is necessary.

There are two things essential for a biometric to be successful in verifying a user:[14]

- The biometric must be unique to that specific individual
- The method used to capture the biometric must preserve this uniqueness so it can be used for proper verification.

In order for biometrics to be used successfully for authentication, each person using the system must enroll in it before use. This involves a user creating some sort of reference template that will be stored within the system for future comparison. When a user then attempts to authenticate to the system, the new biometric capture will be compared against this original reference template to see if they match. Now it is important to note that it is impossible to capture a 100% accurate image or recording of the biometric. Because there could be discrepancies between the reference template and the authentication attempt, we need to focus on capturing a match that is "good enough," rather than one that is perfect. Some sort of threshold must be met before a user can be authenticated. Setting this threshold is extremely important. If you set the threshold too high, valid users will be rejected by the system, creating a high *false non-match rate* (FNMR). If the threshold is too low, invalid users could be accepted, creating a high *false match rate* (FMR).

Now that we have a general idea of how biometrics work, let's look specifically at how fingerprints are used for authentication. Long before computer technology existed, fingerprints have been used to identify who people are. Parents sometimes have their children fingerprinted so in the case they became lost they could be identified. Police always fingerprint criminals or suspected criminals to identify those who commit crimes and to provide evidence that a person was at a crime scene. Art Allan, analyst for Gartner, states "It has been estimated that the chance of two people having the same fingerprint is less than one in a hundred billion…in more than a century of the use of fingerprinting, no two fingerprints have ever been found to be identical."[14] Fingerprinting is definitely a reliable source for authenticating users.

[14] Allan

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

How does today's technology capture fingerprint images for authenticating users? There are several ways that fingerprints can be captured. The first is using an optical device. The finger is placed on a piece of glass or hard plastic, illuminated, and then captured by a charge-coupled device (CCD). For security purposes, the actual image of the fingerprint is not stored. Instead, the pattern of the fingerprint is converted into a template, encrypted, and stored within the system. Unfortunately, optical systems tend to have more problems with fingerprints left by previous users (latent fingerprints). These can interfere with the scanning, and lead to a higher FNMR. Another method used to capture fingerprints is through silicon based systems, which rely on capacitance (the ability to hold an electric charge) to capture a fingerprint image. Because ridges and valleys hold a different electric charge, the reader can distinguish the difference and create an image of the fingerprint. Silicon based systems tend to be smaller than optical systems and are more accurate. Some are even small enough to be used on handheld devices or smartcards.[15] Other technologies, although not used as much, are thermal recognition (distinguishing temperature differences between the ridges and the valleys) and measuring the surface pressure to determine the ridges and valleys in the fingerprint.

Although fingerprint technologies have emerged as the front-runner in the biometric race, there are ways to defeat a fingerprint scanning authentication system. In 2002 a Japanese mathematician, Tsutomu Matsumoto, did some research attempting to defeat both optical and capacitive fingerprint scanners. With a little bit of time and about $10 of supplies from the store, he succeeded in defeating eleven commercial biometric systems![16] Bruce Schneier, renowned security professional, explains how it was done:

> His [Matsumoto] more interesting experiment involves latent fingerprints. He takes a fingerprint left on a piece of glass, enhances it with a cyanoacrylate adhesive, and then photographs it with a digital camera. Using PhotoShop, he improves the contrast and prints the fingerprint onto a transparency sheet. Then, he takes a photo-sensitive printed-circuit board (PCB) and uses the fingerprint transparency to etch the fingerprint into the copper, making it three-dimensional. (You can find photo-sensitive PCBs, along with instructions for use, in most electronics hobby shops.) Finally, he makes a gelatin finger using the print on the PCB. This also fools fingerprint detectors about 80% of the time. [16]

A few other problems exist with fingerprint scanners as well. Fingerprint readers can be subject to environmental problems. If the skin is too dry (for example in health care environments where hands are washed frequently), the image may be misread. In extremely dirty environments, or where gloves must be worn, fingerprint scanners are not the logical choice.

While biometrics can be an excellent technology to authenticate users, they still have some flaws that need to be worked out. While they can be hard to defeat sometimes, they can still be compromised. Once defeated, biometrics cannot be changed. A new finger cannot simply be issued like a password or security token. In addition, there are

[15] Speir
[16] Schneier

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

human acceptance issues that come with biometrics. Some people simply do not want to have to their iris scanned, for example, or are religiously opposed to some types of biometric devices. Because of these issues, tokens and smartcards provide a better alternative for user authentication.

4.2 Tokens – What You Have

Another way users can be authenticated is by forcing them to produce something that only they have. Automated Teller Machines (ATM) are a great example of this. In order for a user to access their account, they must insert *their* card into the machine (in addition, a PIN is required to further verify their identity). Without a card, they cannot access their accounts. Another example is a building access card. Without the card, a user will not be granted access to enter the building. Similar devices have been developed for the information security market to protect access to critical computer systems. This paper will briefly explore how RSA SecurID tokens are used to authenticate users.

RSA SecurID

RSA Security is the leader in strong authentication token technology. Their product, SecurID, comes in multiple formats: hardware devices such as keyfob tokens and cards similar in size to a credit card, or in software formats for use on Palm, Pocket PC, and other mobile devices. SecurID greatly strengthens authentication because it uses *two-factor authentication* – users are authenticated based on two factors: something they have (the SecurID device) and something they know, a PIN number. When a user logs on to a system that requires SecurID, they will enter their userID and a passcode consisting of their PIN+the code displayed on their SecurID token. This code changes every 60 seconds, so in essence a user's password changes every minute! In addition, the code can only be used once (this is called a One Time Password – OTP) so even if the code was captured in transmission, it could not be re-used!

How does the RSA SecurID generate this random code? Every SecurID token that is shipped from the factory is initialized with a unique seed value. Every minute, the internal chip performs an algorithm that scrambles this unique seed value with the current time to produce the token code. The server, which knows the seed record for each token, performs this same algorithm. If the passcodes match, the user is authenticated. SecurID uses the standard AES algorithm to ensure the quality and integrity of the encrypted time/seed hash that is sent to the server.

To make sure SecurID works across differing time zones, all SecurID products are synchronized to Universal Coordinated Time (UCT), which is the same as Greenwich Mean Time (GMT). In addition, the server compensates for clock drift by accepting passcodes that are valid within a three-minute window. The server will first check to see if the username and PIN are correct. If so, it will check to see if the passcode matches that of the current minute. If it does not, the server automatically checks the

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

previous and subsequent minutes to see if the passcode matches. If it does, the user is authenticated and a note is made in the user's record stating the server needs to adjust future login attempts for this clock drift. As long as a user logs in regularly, their token should never drift outside of this three-minute window. If a passcode falls outside of this window, the server will attempt passcodes for the 20 minutes in front of and behind the current minute. If it finds a match, the server asks the user for the next tokencode to verify that the user actually has the token. If the clock drift matches, the server authenticates the user and notes the clock discrepancy in the user's record.[17]

RSA has also recently partnered up with Microsoft to create a product called 'SecurID for Microsoft Windows.' This product integrates directly with Windows domain controllers and Active Directory. Once authenticated, the RSA server passes the user's windows password to the system, which is given to the domain controller to finish the authentication process and deliver the completed kerberos ticket.

SecurID for Windows allows users to authenticate to Windows sessions using a standard SecurID token even if the user is offline. To accommodate offline access, once the user is authenticated online, the RSA server will send the client pre-calculated codes for a certain number of days. When logging in offline, the client checks against these codes, rather than against the server. If properly authenticated, it will then pass the user's local password to the system instead of the domain controller to complete the authentication process.

This system also easily accommodates system password change policies. When a user's Windows password expires, the user is prompted with a window with their old password box already filled in. They can then change their password to comply with policy, and then login with SecurID. Once authenticated, SecurID handles all the windows passwords in the background so the user doesn't have to worry about it! All the user must do is remember their PIN and take their token to work with them![18] At the writing of this paper this product was still in beta testing.[19]

Although SecurID can be used to greatly improve authentication security, there are still some limiting factors that must be considered before implementation. The first is cost. Passwords are cheap to implement. RSA SecurID, however, has up-front costs of purchasing the system, the tokens, and the cost of deployment. In addition, each time a new user is setup, coordination must be done to deliver a hardware token to that user, possibly delaying their productivity. The SecurID tokens also have a shelf life. At a time specified on the back of the token, the token will no longer work, and will need to be replaced. This can cause a headache because new tokens must be purchased and re-deployed. This especially becomes an issue when tokens are issued to third-party users that may not be employed by your company or may not be in close proximity to your location. Some RSA tokens also are susceptible to static electricity, causing them to fail. Because of this, processes must be put in place for replacing broken tokens.

[17] The Power Behind RSA SecurID Two-Factor User Authentication: RSA ACE/Server
[18] RSA SecurID for Microsoft Windows
[19] Microsoft, RSA Begin Testing Security Tokens

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Even though some limitations exist, however, the security gained by two-factor authentication easily outweighs them. In addition, much (if not all) of the up front cost will be re-gained in reduced helpdesk support costs for password resets. Finally, the risk of attacks via stolen passwords is mitigated, saving thousands of dollars that an otherwise successful attack could cost to clean up.

5.0 Conclusions

Passwords are still ubiquitous for user authentication in business today. This is mainly because they have a low up-front cost and are easy to implement. However, there are many problems with passwords that cause them to be easily compromised. Because passwords are simply character strings, they are susceptible to guessing attacks, based on either social information gained from the user or brute force attacks. Passwords are also subject to password cracking attacks if attackers can obtain the system password file. Social engineering is increasingly being used to harvest user passwords so attackers can illegally enter protected systems. Because of these many flaws, passwords should be avoided if possible when protecting critical information. If passwords cannot be avoided, strong passwords should be required. Passwords should be required to be at least eight characters long, with multi-case letters, numbers, and special characters. Users should be encouraged to build passwords based on phrases rather than words. Passwords also should be required to be changed regularly to mitigate cracking attacks.

When possible, strong authentication should be used instead of passwords. Biometrics are a better alternative to passwords, but there are still many flaws associated with the technology. Because of these flaws, security tokens are a better alternative for strong authentication. RSA SecurID is an excellent product that mitigates almost all problems associated with passwords. We must protect our sensitive information by using a layered approach. Without multiple layers of security, our information is ripe to be harvested by attackers. Using tokens for strong authentication is an excellent means to secure the authentication process and protect our userIDs and passwords from those working to steal our information.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

6.0 References

Allan, Ant. "Something You Are (or Not): The State of Biometrics." IT Security Summit 2004. Gartner. Marriott Wardman Park Hotel, Washington, D.C. June 9, 2004.

"authenticaton." SearchSecurity.com. 10 Nov 2003. http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci211621,00.html (24 Jun 2004).

Beverstock, David. "Passwords are DEAD! (Long live passwords?)." 5 Jul 2003. http://www.sans.org/rr/papers/index.php?id=1144. (1 Jun 2004).

Granger, Sarah."The Simplest Security: A Guide To Better Password Practices." 17 Jan 2002. http://www.securityfocus.com/infocus/1537 (30 Jun 2004).

"How to disable LM Authentication on Windows NT." 1 Dec 2003. http://support.microsoft.com/default.aspx?scid=kb;en-us;147706 (28 Jun 2004).

"How to enable NTLM 2 authentication." 5 May 2004. http://support.microsoft.com/default.aspx?scid=kb;en-us;Q239869 (30 Jun 2004).

Kabay, M.E. "The End of Passwords: Inadequate Solutions." Network World Newsletter. 1 Jun 2004.

"LC5 Datasheet". 2004. http://www.atstake.com/products/lc/acrobat/lc5_datasheet.pdf (28 Jun 2004).

"LC5 Feature Comparison Chart". 2004. http://www.atstake.com/products/lc/features.html#features (30 Jun 2004)

"Microsoft, RSA Begin Testing Security Tokens." Security Pipeline. 3 Jun 2004. http://www.securitypipeline.com/showArticle.jhtml?articleID=21401314 (1 July 2004)

Mogull, Rich. "Don't Catch a Social Engineering Disease." IT Security Summit 2004. Gartner. Marriott Wardman Park Hotel, Washington, D.C. June 8, 2004.

"password cracker." SearchSecurity.com. 29 Apr 2004. http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci536994,00.html. (28 Jun 2004).

"Passwords revealed by sweet deal." BBC News, UK Edition. 20 Apr 2004. http://news.bbc.co.uk/1/hi/technology/3639679.stm (19 Jun 2004).

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

"RSA SecurID for Microsoft Windows." RSA Security Whitepaper. 2004. http://www.rsasecurity.com/products/securid/datasheets/SIDMS_DS_0504.pdf (7 Jul 2004).

Schneier, Bruce. "Fun with Fingerprint Readers." Crypto-Gram Newsletter. 15 May 2002. http://www.schneier.com/crypto-gram-0205.html (7 Jul 2004).

Shimonski, Rob. "Hacking Techniques: Introduction to password cracking." 1 Jul 2002. http://www-106.ibm.com/developerworks/library/s-crack/ (28 Jun 2004).

Speir, Michelle. "Atmel solves fingerprint riddle." Federal Computer Week. 18 Aug 2003. http://www.fcw.com/fcw/articles/2003/0818/tec-review-08-18-03.asp (7 Jul 2003).

"The Power Behind RSA SecurID Two-Factor User Authentication: RSA ACE/Server." RSA Security Whitepaper. 2003. http://www.rsasecurity.com/products/securid/whitepapers/AS51_SB_1103.pdf (7 Jul 2004).

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.