Global Information Assurance Certification Paper

Beyond Passwords
Advances in User Authentication

Joshua Frankel
January 4, 2004

Abstract

Network security is typically defined as that which ensures confidentiality, integrity, and availability[1] on a network. This is accomplished at a high level through linked authentication and authorization systems. Before a user may utilize a resource on a secured network, they must prove their identity to the network's authentication agent. If they are successfully authenticated, the network will consult the authorization subsystem to determine what systems that user may access.

For years, usernames and passwords have been considered sufficient for the majority of user authentication scenarios on a network. When authenticating to a domain, website or hardware device, frequently the only means of restricting access are to limit the source of the connection and increase the complexity and number of passwords required to gain access. While this will certainly prohibit casual users from authenticating and gaining access, many issues remain with relying solely on static passwords. This paper will investigate the growing needs for strong authentication, discuss the weaknesses associated with relying on passwords for user authentication and will examine the alternative authentication methods that are in use today such as key fobs, fingerprint scans, and more novel methods.

One of the primary goals of a network is to provide resources to authorized users. Users are defined by their identity – literally the name by which they can be recognized[2]. This often takes the form of a user account. Based on this identity, the network can provide services such as:

- **Authorization**: Access or authorization systems are frequently cross referenced with a list of permitted identities
- **Personalization**: Applications can provide unique views and presentations depending on the identity
- **Auditing**: Systems that track usage reference their data to an identity to leave a record of who did what, and when they did it.[3]

An essential element of security in this process that is often overlooked is authentication. This is the act of reliably verifying the identity of someone (or something) or, more specifically in the network security realm, it is the process of verifying that a requester has been issued a unique identifier and knows the secret that is associated with that identifier.[4] Without authentication as the

[1] http://slis-two.lis.fsu.edu/~security/SecurityGlossary2.html
[2] Cole (p. 298)
[3] http://www.rsasecurity.com/products/authentication/whitepapers/ASC_WP_0403.pdf
[4] http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0266.html?Open

precursor of using the identity, there is no guarantee that the active user of the identity is who they claim to be.

Historically, at its most basic level, user authentication has been accomplished through a username and password combination. Each user has a unique username and password associated with their identity. When a need for authorization is required, the user inputs the correct username and password. If no restrictions are placed on users, 80% or more will use simple passwords that are easily broken or guessed such as their birthdate, the name of a relative or a famous person.[5] Some users will even use their username or no password at all! I frequently tell the tale of a prospective client who was a government contractor on a secure installation. Their sole domain controller and file server was on a table in the corner of the lobby all setup with a keyboard, monitor, mouse… and a blank administrator password. Since they had been running this way for over a year without incident, they saw nothing wrong with this configuration.

To provide additional security, the password (or username, which can be considered another password in some instances) can be made more complex by requiring minimum lengths, changes in character case, utilizing numeric or symbolic characters, frequent changes, or any combination of these. Multiple levels of passwords are often required to progress to additional levels of security. Many security implementations will stop at this point and consider these concentric rings of complex password challenges sufficient to provide validation of a user's identity. In reality, all these can do is slow a determined attacker.

All passwords are vulnerable to attack from a number of vectors. The simplest is the brute force attack. This consists of methodically trying different passwords until one successfully authenticates the attacker. The more complex the password, the more time and / or computational power are required to crack it. Many authentication systems are configurable to lock an identity after a preconfigured number of invalid password entries or to force delays between successive attempts, but often these options are not enabled by default. Even if they are enabled, a determined attacker can capture a system's password file, and with the right tools, crack every password within on their own system, limited only by the speed of their hardware and quality of their tools. One of the best and most commonly used utilities for Windows password cracking is L0phtCrack by @stake. They claim that "at one of the largest technology companies, where policy required that passwords exceed 8 characters, mix cases, and include numbers or symbols, L0phtCrack obtained 18% of the passwords in 10 minutes, 90% of the passwords were recovered within 48 hours on a Pentium II/300 and that the Administrator and most Domain Admin passwords were cracked."[6] Allowing an attacker to compromise a password file obviously is indicative of other serious issues with a network's security, but it still demonstrates how poor a method of authentication passwords can be.

[5] http://www.infosatellite.com/news/2002/10/a071002windows_flaws.html
[6] http://www.atstake.com/products/lc/

In an attempt to mitigate the damage that a compromised password can cause, security policies frequently mandate regular password changes. Ironically, these policies often work against administrator intentions and create further security holes. Many users will attempt to rotate between the same few passwords, meaning that if one is compromised, eventually an attacker will be able to use it even after it has been changed. If users are forced to create new passwords at every mandated change cycle, often they will be unable to remember them, and will write them down somewhere near their workstation. This is catastrophic for administrators since anyone that gains physical access to the user's workspace can masquerade as that user and defeat the network's authentication challenges.

Often the simplest form of attack is through social engineering. This consists of "clever manipulation of the natural human tendency to trust"[7] in order to gain information or access to protected environments. This can be as simple as calling a corporate help desk and pretending to be a user who has forgotten their password. Other common techniques are to contact the user while posing as an administrator and to ask them for their password. Social engineering is often the fastest and most effective way to compromise any protected environment, and is extremely difficult to protect against without strict training and user policies. To quote the infamous security specialist Kevin Mitnick, "You could spend a fortune purchasing technology and services . . . and your network infrastructure could still remain vulnerable to old-fashioned manipulation."[8] If a password is gained in this manner, it can be extremely difficult to detect since a well-executed social engineering attack should not raise any suspicions or show any signs of forced entry. The attacker will be able to assume the user's identity with impunity until the password is changed. For this reason as well, it is extremely dangerous to rely on passwords as a sole source of authentication.

The inherent weakness of passwords as a sole means of user authentication is that they are only one method, or factor, to base authentication upon. Typically, there are three factors that can be considered as a means of authentication:

- **Something you know**: Most commonly this is a password or personal identification code such as a social security number
- **Something you have**: This is frequently a smart card or card key. It can also be a one-time pad or challenge-response list or even a digital certificate
- **Something you are**: Biometric solutions use this authentication factor by reading in fingerprints, retinal patterns, or voiceprints[9]

None of these factors are inherently more secure than the others – they all have their own strengths and weaknesses. It is in the combination of factors that

[7] http://www.securityfocus.com/infocus/1527
[8] http://www.securityfocus.com/news/199
[9] http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0266.html?Open

systems can move toward achieving reliable authentication. It can be relatively easy to compromise a scheme based on only one of these factors, but to compromise multiple at once raises the degree of difficulty by orders of magnitude. An example of single vs. multiple factor authentication is a system that relies on card keys completely for authentication. Even if the holder of the card is not the rightful owner, they can authenticate to the security systems and gain authorization just as if they were until the card is revoked. This means that if a card is lost, whoever recovers it has that user's full access. An employee could use a co-worker's card without the rightful owner or the security system being aware. If another authenticating factor such as a passcode or fingerprint scan is required in addition to the card, possession of the card is useless without that other factor. In this way, authentication methods that are not sufficient on their own can become more resilient. All methods still have tradeoffs and their own strengths and weaknesses. At this point, there is no single authentication scheme that cannot be deceived – it is only a question of the difficulty involved.

One of the most common ways for an organization to implement two-factor authentication is with RSA's SecurID. This third-party system consists of a PIN code (comprising the "something you know" factor) and a SecurID Authenticator (usually a smart card or key fob, though it can be software based) that displays a new number (usually six digits long) every minute for the "something you have" factor. The number displayed, referred to as the tokencode, appears to be completely random, but is actually generated by combining a hard-coded seed value (which is unique to each token) with the current time through a confidential algorithm. An authentication server running the RSA ACE/Server software can independently derive the currently displayed tokencode on any SecurID device so long as the server knows the current time and the initial seed value for the device. It is impossible to predict the next tokencode that will be displayed without knowing the algorithm and seed value (which are never publicly available), even if previously displayed values are known.[10] Once a particular passcode is used, it cannot be used for a second time in the minute that it is valid. This prevents replay attacks by capturing the passcode and replaying it to the ACE Server.

Authentication takes place through a RSA ACE/Agent that must be installed on the point of authentication (i.e. Desktop, web server, VPN device, etc). The user is prompted to authenticate to the network by entering a passcode in addition to their standard username / password login. This passcode is comprised of the PIN appended to the currently displayed tokencode. The code is passed from the ACE Agent onto the ACE Server (or server replica) where it is cross-referenced with the username. The server has a record of what tokens have been issued to the user, what values they should display at the time of the authentication request, and what the user's PIN is. Based on whether the passcode entered matches the expected code, the ACE Server sends a response back to the ACE Agent either accepting the authentication or denying it. After a certain number of

[10] http://www.rsasecurity.com/products/authentication/whitepapers/ASC_WP_0403.pdf

denied attempts, the server will not accept any more requests from that token until an administrator re-enables it.

The benefits of this architecture are many. If deployed properly, it can condense the authenticators required for many checkpoints into one tokencode that will work for all of them. There are ACE Agents for a number of different platforms, including Windows, Linux, UNIX, Netware, IIS and Apache[11]. There are also over 230 products that are SecurID Ready out of the box[12] along with any with built in RADIUS and TACACS+ support. This ensures that SecurID will integrate well into the majority of environments. SecurID uses a constantly changing passcode, which requires a user to be in possession of the token in order to authenticate – if an attacker merely glanced at the display, the value would only be valid for a short period of time and they would still need the secret PIN.

Despite all of these advantages that SecurID (and other similar two-factor authentication models) have over standard passwords, they still have Achilles heels that can be attacked. Desktops, laptops or servers that have the ACE Agent installed locally often are configured with a Reserve Password. This is a backdoor past SecurID authentication that allows access to the system with just a static password. Often this is setup in case the ACE Agent is unable to communicate with an ACE Server to validate the passcode, or if the administrator does not have their token for some reason. These reserve passwords are rarely used in normal practice, and are rarely updated. Often to ease the management responsibilities of administrators, these passwords are the same on all servers and / or client machines. This means that if a reserve password is compromised, not only would the attacker gain authentication and access to a protected machine, but they could also potentially authenticate to any other host with the same reserve password. This is a frequently seen problem with authentication schemes that heavily rely on factors other that "something you know" – administrators (and sometimes users) have occasional needs to get into systems without the requisite factor. To facilitate this process, a backdoor password is often permitted. While useful in certain situations, this runs the risk of compromising the security of all protected resources.

Another similar vector of attack is another feature relating to a lost token. In the event that a user loses or breaks their token and the resources they need to authenticate to either do not have a reserve password or the administrator is unwilling to supply it to the user, the ACE Server can be configured to allow the user to use a list of one-time use tokencodes which it generates, then gives to the administrator to pass on to the user. While this is an implementation of a one-time pad and is still extremely difficult to attack with brute force or cryptographically, it is far less secure. If the paper or electronic document containing the tokencodes is compromised, the attacker can copy it and use the codes themselves. This is also an area that is ideal to a social engineering

[11] http://www.rsasecurity.com/products/securid/specs/agentsupport.html
[12] http://www.rsasecurity.com/products/securid/rsaaceagents.html

attack. A would be attacker can easily contact the administrators posing as a user who has lost their token. If convincing enough, the administrators might supply them with the one-time keys, which would eliminate the need for physical possession of the token. They would still need to know the user's PIN, however. That is another potential vector of attack.

The PIN is just another type of password, and should be treated as such. It is usually four to eight characters long and can be alphanumeric. Mechanisms exist in the ACE Server to force PIN changes at set intervals. This comes back to the issues associated with frequent password changes – the user will often choose simple, easy to remember (and guess) passwords or will write them down. Since authentication requires possessing the token at every challenge, users will frequently keep the PIN with the token. I have even seen environments where some users kept track of their Pin by taping it onto the back of their token! This, of course, destroys the concept of two-factor authentication. With the PIN written down and stored with the token, all that an attacker needs is physical possession of the token, ("something they have"), and they can masquerade as that user's identity.

Another issue with SecurID is common to most other enhanced authentication schemes. All points of authentication need to be protected by the software agents in order to accept and process the enhanced request and pass it on to the server. In environments with non-standard hardware devices and / or software (including custom in-house applications) there often are no ACE Agents available. This requires administrators to either create their own with RSA SecurTools[13] or to contract with RSA to develop the Agent for them. Either way requires a significant time and financial commitment which businesses are often unwilling to expend on enhanced authentication for all but the most critical applications. This provides a point of entry to the environment with less security than those protected with the Agent. Administrators will also often use SecurID or systems resembling it to protect external access such as VPN, Extranets, or RAS while relying on standard passwords for internal resources such as desktops. This is often to reduce administrative overhead and / or costs in the belief that attackers will be attempting to penetrate the system from the outside. All a would-be attacker needs to do to circumvent these protections in such an environment is to compromise an internal computer, (such as emailing a Trojan horse that permits remote connections to an infected desktop to an unsuspecting gullible user), and they will be able to access the network remotely with only the standard username/password. It is critical to remember when deploying any security solution that the system is only as strong as its weakest link. If there are authentication checkpoints that are not protected by strong two-factor authentication, these will likely be the focus of attack by those looking to gain unauthorized access.

[13] http://www.rsasecurity.com/services/app_dev/agents-securtools.html

The third type of authentication factor is “something you are”. This is covered by the growing field of biometrics – which “refers to the automatic identification or identity verification of living persons using their enduring physical or behavioral characteristics”[14] Often this will take the form of a fingerprint or handprint scan, though there are many other personally unique identifiers that may be used. For years these have been marketed as the authentication solution of the future[15] without considering the new and existing problems that persist with schemes built around biometric authentication. Two-factor authentication schemes that rely on a PIN or password in addition to the biometric scan are more secure, but are vulnerable to issues previously discussed such as users either never changing or conversely, writing down their PIN as well as others.

The first issue to address is how to integrate the authentication method with the existing environment. The key for a successful biometric authentication scheme is to “the ability to seamlessly integrate into existing authentication mechanisms which have no standard implementation.”[16] Since biometric authentication is still a developing industry, there is no clear standard for all devices and authenticators to follow. Microsoft has been making progress in this by integrating smart card authentication based on digital certificates[17]. This is a solution for smart cards primarily (“something you have”), but it can be adapted for use with fingerprint readers and other biometric solutions. The underlying technology has not significantly advanced in the past few years. It still requires complex procedures to initialize the cards and distribute them to users. Biometric solutions are completely dependant on third party products to extend the base Microsoft certificate authentication. In UNIX or Linux, PAMs (Pluggable Authentication Modules) are used to extend the base system architecture to accommodate new authentication schemes, while Netware uses their own NMAS (Novell Modular Authentication Service) product to extend NDS to accommodate biometrics[18].

All of these solutions provide only frameworks of varying quality - NMAS is the most developed and provides services for passing the authentication token to an Active Directory domain as well as containing provisions for graded authentication which allows for different levels of authorization dependant on the method of authentication, i.e. a password may allow you to connect to a share, but you can only get at a particular subdirectory with a fingerprint scan. To extend these frameworks to allow for biometric authentication, third party software and hardware are required. Often this equipment needed to implement biometric authentication is significantly more expensive upfront than for other authentication methods, and the majority of organizations are not willing to spend any more than they absolutely have to on authentication. This leads even some

[14] http://www.eff.org/Privacy/Surveillance/biometrics.html
[15] http://www.cnn.com/2001/TECH/computing/01/17/biometrics.future.idg/
[16] Hsu (p. 17)
[17] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/smrtcard/scard.asp
[18] http://www.networkcomputing.com/1403/1403ibg122.html

of the security conscious corporations to rely on authentication by other methods such as smart cards or even passwords alone.

Another major issue is accuracy. No biometric system is able to say with 100% certainty that the requestor's biometric signature matches the one stored in the identity database. All systems have a margin of error that must be taken into account. There are two ways that a biometric system can fail – by a false match (incorrectly matching a subject with someone else's reference sample) and a non-match (failing to match a subject with her own reference sample).[19] Some systems allow the sensitivity to be configured by the administrators to decrease the odds of the non-match. This leads to one of the constant battles of security – tight security vs. convenience. If a biometric authentication system is too sensitive, it will generate a higher number of false denials. If this becomes too frequent, users will become frustrated. If, to appease the users, it is geared lower, this could allow unauthorized authentication. There already are ways of fooling biometric devices. The Japanese cryptographer Tsutomu Matsumoto has developed a way of generating false matches on biometric fingerprint readers using commonly available tools and $10 of common household supplies.[20] He demonstrates a way of lifting a latent fingerprint, and imprinting it on a gelatin mold that is placed over the fingertip. He claims to be able to circumvent eleven different commercial fingerprint readers, including those with optical, capacitive, and "living finger" sensors.[21] Retinal scanners have been shown to be vulnerable to contact lenses, and voiceprint scanners have been defeated by playing back voice samples.

A previous technique for bypassing biometric scanners was to remove the item to be scanned from the rightful owner, but modern systems have mechanisms to detect if the "token" is attached to a living human. In the even that an authorized user is unable to authenticate due to the loss of the biometric item, some systems allow for alternate biometric measurements (more expensive and hardware intensive) while others allow a password to be used as a backup (similar problems as discussed previously – this is an Achilles Heel of the system). This technology is improving, but it is currently far from perfect. At this point, organizations should always use a second factor such as a password along with any biometric authentication point to reduce the chance of false positives and to make an actual attack more difficult. Biometric authentication, or likely any authentication based around a single factor, should not be the sole basis of authentication.

There are many new methods of authentication being devised all the time. Rachna Dhamija describes a system of authentication based on image recognition.[22] He proposes a system where users choose from a selection of

[19] http://www.eff.org/Privacy/Surveillance/biometrics.html
[20] http://www.schneier.com/crypto-gram-0205.html#5
[21] http://www.schneier.com/crypto-gram-0205.html#5
[22] http://www.ece.cmu.edu/~adrian/projects/usenix2000/usenix.html

non-photographic, random images and add them to their personal profile. When prompted for authentication, they are shown a range of images and asked which ones are chosen from their profile. If they correctly answer, they are authenticated. This system prevents users from writing down their passwords, and prevents them from easily telling anyone what it is. It also does not require any additional hardware to be distributed to the users. The issues are the initial training time required to select the images for the profile, and the difficulties in users remembering which images they selected. The initial phases of implementation of this project could be difficult as users struggle to remember which images they selected for their profile.

Another novel proposed technique is based around mouse gestures. Researchers at Queen Mary University of London have devised a method of identifying people by the way that they sign their name with their mouse.[23] It analyzes the range of motions used to establish the signature, and then compares them for authentication. Few details have been released, but this appears to be a very weak scheme. A "mouse recorder" can be installed to capture a valid user's movements, which can be used to replay this later by unauthorized users. Non-match failures should also be common due to differences in mice used, seating, workspace, and other variables.

There are many different methods of authentication available today, but none offers a completely foolproof solution on its own. The most reliable method of proving the identity of a user is still to use multiple forms of authentication. Though each form has its own individual weaknesses, when used together the odds of false authorization decrease. The defense-in-depth concept applies with authentication in that more (well constructed) layers lead to tighter security. So long as these layers are not so arduous for the users that they "revolt" and work to circumvent the systems, they will ensure that only authorized users are able to access the protected resources.

[23] http://www.findarticles.com/cf_dls/m0ECZ/2003_Sept_2/107166243/p1/article.jhtml

References

Dean, Jason. "Internet Security Glossary." July 16, 2000. URL: http://slis-two.lis.fsu.edu/~security/SecurityGlossary2.html (January 10, 2004).

Cole, Eric. Fossen, Jason. Northcutt, Stephen. Pomeranz, Hal. SANS Security Essentials with CISSP CBK. SANS Press, 2003. p. 298.

RSA Security. "The Authentication Scorecard." August 15, 2003. URL: http://www.rsasecurity.com/products/authentication/whitepapers/ASC_WP_0403.pdf (January 10, 2004).

IBM. "The Need for Authentication and Authorization." August 1, 2003. URL: http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0266.html?Open (January 10, 2004).

Sigvartsen, Ana Letícia. "The 10 Most Unwanted: Windows security flaws." October 07, 2002. URL: http://www.infosatellite.com/news/2002/10/a071002windows_flaws.html (January 10, 2004).

@Stake. "About LC 4." 2004. URL: http://www.atstake.com/products/lc/ (January 10, 2004).

Granger, Sarah. "Social Engineering Fundamentals, Part I: Hacker Tactics ." December 18, 2001. URL: http://www.securityfocus.com/infocus/1527 (January 10, 2004).

Mitnick, Kevin. "My first RSA Conference." April 30, 2001. URL: http://www.securityfocus.com/news/199 (January 10, 2004).

RSA Security. "RSA ACE/Agents Platform Support Matrix." 2004. URL: http://www.rsasecurity.com/products/securid/specs/agentsupport.html (January 10, 2004).

RSA Security. "RSA ACE/Agent Software." 2004. URL: http://www.rsasecurity.com/products/securid/rsaaceagents.html (January 10, 2004).

RSA Security. "RSA ACE/Agent Software." 2004. URL: http://www.rsasecurity.com/services/app_dev/agents-securtools.html (January 10, 2004).

Abernathy, William. "Biometrics – Who's Watching You?" URL: http://www.eff.org/Privacy/Surveillance/biometrics.html (January 10, 2004).

Fonseca, Brian. "Biometrics scan the future of security." January 17, 2001. URL: http://www.cnn.com/2001/TECH/computing/01/17/biometrics.future.idg/ (January 10, 2004).

Hsu, Andrew. "Fingerprint security systems come of age." Portable Design. Volume 9 (2000): 17.

De Clercq, Jan. "Smart Cards." URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/smrtcard/scard.asp (January 10, 2004).

Fratto, Mike. "Are biometrics the answer?" February 20, 2003. URL: http://www.networkcomputing.com/1403/1403ibg122.html (January 10, 2004).

Schneier, Bruce. "Fun with fingerprint readers." Crypto-Gram Newsletter. May 15, 2002. URL: http://www.schneier.com/crypto-gram-0205.html#5 (January 10, 2004).

Dhamija, Rachna. “Déjà Vu: A User Study Using Images for Authentication.” September 1, 2000. URL: http://www.ece.cmu.edu/~adrian/projects/usenix2000/usenix.html (January 10, 2004).

“New technology could enable online signatures via the mouse.” September 2, 2003. URL: http://www.findarticles.com/cf_dls/m0ECZ/2003_Sept_2/107166243/p1/article.jhtml (January 10, 2004).