

Security Standard for Authentication

Information Technology Security Standard #2, v1.2

Standards Name: Authentication Standard (a.k.a the password standard)

Original Release Date: 4/5/2006

Latest Revision Date: 8/10/2007

Authority: This standard was approved by the Information Security Council and the CIO, and is supported by the Technology Leadership Committee (TLC). It derives its authority from the CIO and the campus Information Technology (IT) community.

Summary: This standard provides the security requirements for people who wish to gain access to an IT service, which is a process known as Authentication.

Applicability/Scope: This applies to all University owned and operated IT systems and services and to any research or commercial system that offers services to the campus or the Internet. It is not required for privately owned devices or other systems intended for personal use but is recommended.

Responsible Office: ITS/CIO.

Reason and Background for this Standard

Authentication is the process by which a person or a machine provides an identity and proof of that identity to an IT system or service. These credentials are most frequently implemented as a user id and a password. Since authentication is the primary way that user and administrative access is granted to IT systems and services, it is a critical part of system security. Systems with weak or no authentication controls are frequently compromised.

Authentication only provides proof of identity; it does not directly provide the level of access, known as authorization, although a simple mapping from authentication to authorization is common.

There are different authentication requirements for different situations, which are dictated by the sensitivity of the service, the security threats, and the cost of implementation. Typically, application and system administrators with full access to a service will require more rigorous authentication checking than end users. In addition, many IT Systems have both user services and management interfaces and each has different authentication requirements. For the purposes of this standard, we use the following definitions:

- **User Authentication/Access:** access to the application or service by end users who have no special rights on the system or ability to change the underlying system configuration. This is what most users think of when they “log on” to a system or service.
- **Administrative or Management Authentication/Access:** access to an application, system, or service that allows full or extensive control, such that configuration changes can be made or that allows changes to the security configuration of databases, applications, services, or underlying operating systems.

Distributing Credentials

An often overlooked issue with credentials is the methods by which they are distributed. As a set of credentials provides increasing access to sensitive information or systems, it should be distributed with increased caution and verification. Using e-mail may be acceptable when distributing a new password which has limited access to a service with relatively low sensitivity, but it is not suitable for distributing more powerful passwords. In some cases, proof that the intended individual received the set of credentials is necessary and hand delivery may be the only safe option.

Password Complexity (Strength)

A common point of discussion is what the minimum requirements for passwords should be. Countless pages have been written on this subject, and while agreement is not total, there are some common results that should be applied when creating passwords:

1. People cannot typically remember more than seven to eight characters in the typical password format.¹ Pass phrases consisting of many words allow for longer passwords, but are not implemented on most systems and still must be chosen carefully to gain any real strength against attacks.
2. Short passwords consisting of only alphanumeric characters are easily attacked, and are not suitable for controlling access to any service of even minor importance. Such weak passwords should only be used when intended to provide a minimal defense against automated attacks.
3. Passwords of less than seven characters are nearly useless. Passwords that combine dictionary words with a suffix/prefix of numbers or symbols are almost equally worthless. “#orange123” is little better than the all-time favorite of “orange.”
4. Simple substitutions, such as @ for A, ! for ell, etc., are well known to password crackers and not very helpful in increasing password security.
5. The NetID password is limited to 8 characters as a maximum and we therefore have to use a large character set to keep it strong. For those systems that can use longer passwords, the character set can be simplified, but this means using 10 characters or more.
6. More detailed information about password strength is covered in Appendix A.

Default and Missing Passwords

System administrators need to be aware that the default passwords that come with most systems and applications (which often default to NO password) are nearly worthless and must never be used. These passwords are gathered into widely disseminated lists on the Internet and many hacker tools incorporate the lists. Newer exploits can even play tricks with the user's own web browser to attack their home routers when default passwords are in place.

Changing Passwords

An additional point of contention is the question of how often passwords should be changed. Requiring frequent changes will cause great overhead and frustration for users and system operators alike, while never requiring a change will leave compromised passwords available to attackers for lengthy periods. A general rule-of-thumb within the edu space is that once a year is a good balance. It is generally accepted that a good, strong password that changes relatively infrequently is better than a somewhat weaker password that is changed regularly. The best solution is a system that uses one-time passwords, such as those incorporated into tokens.

Password Lockouts

Another common question surrounds the idea of locking an account temporarily after a series of failed login attempts. The idea is that someone attempting to brute-force guess a password will have many failures while they guess, and a lockout slows their ability to complete the attack. This is largely correct, but is often coupled with misconceptions about the likelihood of gaining access with this kind of attack. Even a six character alphanumeric password has $62 \times 62 \times 62 \times 62 \times 62 \times 62 = 56$ billion combinations. If we allow 10 guesses before a lockout, then lockout for 10 minutes, the attacker can only try about 500,000 passwords per year. (A far more productive attack is to sniff or gain the password "hashes" and then use direct attack tools. Or better yet, use social engineering tricks to get someone to reveal their password.) A general recommendation for lockout is to execute the lockout after 10 successive login failures, and to leave the lock in place for 10 minutes. Monitoring for many lockouts per minute is a good way to detect password-guessing attacks. Keep in mind that an attacker can simply try random passwords against all your accounts at a high rate and create a denial-of-service condition for your users (and this has happened on campus), so use lockouts judiciously when providing authenticated services over the Internet or within Windows domains.

Protecting Passwords

The final issue is the storage and transport of the password, or a password equivalent such as a password hash. Sniffing and keyloggers can easily obtain passwords or their derivatives and allow for easy cracking and access to accounts. Use of cleartext passwords, which are inherent in protocols such as ftp and SNMPv1, is an extremely dangerous practice and easily solved by using replacements such as sftp (which means

either ftp using SSH techniques, or ftp coupled with SSL), SSH/SCP, SNMPv3, etc. Many campuses have banned all use of cleartext passwords on the wire.

Stored passwords, which are usually stored as hashes or even as encrypted versions, also need to be carefully protected with proper access permissions. Cleartext passwords that are embedded in applications and services are another serious risk, particularly if the server is accessible by many people and the stored password is not well protected. Such embedded passwords need to be changed according to the same rules as any administrator-level password.

Authentication Standard – User Access – All Covered Systems and Devices

1. Unless a service is intended to provide public or anonymous access, and has a legitimate need to do so, all users of the service *must* authenticate by providing a set of credentials. If the service provides access to any non-public University Data, then user authentication is *required*.
2. Each User of an IT service *must* be represented by a unique identifier (user name) and associated proof of identity (e.g., password). These credentials may be reusable (e.g., static password) or variable (e.g., one-time password).
3. Users *must not* share credentials, except in unusual circumstances where immediate access is required to avoid disruption of University business.
4. Authentication processes *must* protect the credentials from interception in a form that could be reused to impersonate the user. Weak credentials, such as cleartext passwords, *must* never be used over insecure network links without adding encryption.
5. All authentication successes and failures *must* be logged and include login/logout times.
6. Wherever possible, the system should rely on a centrally-provided and provisioned authentication source, rather than using local account databases. Central authentication services ensure more rigor in assigning and removing accounts and credentials and make auditing for security problems more cost-effective. When a local account database is used, the system operator is responsible for requiring passwords at least as complex as the NetID standard, and for timely deprovisioning of unnecessary or obsolete accounts. “Timely” will reflect the local environment and processes that surround the system and is subject to review by the ISO.
7. Any system that provides anonymous or guest access must document the types of access provided, assess the risks posed, and ensure that a process is developed for regular review and removal of unneeded guest accounts.

The current complexity requirement for NetID passwords is:

- 7 or 8 characters in length (this limitation is a function of the older UNIX system).
- Includes at least one uppercase and one lowercase letter, one or more digits, and one or more non-alphanumeric symbols. Unfortunately, the list of symbols is limited by the backend systems to include: ` ! # \$ % & * () - _ = \ | [] ' ; : / ? . ,

- Does not include any dictionary word of three characters or more or any common name.

Authentication Standard – Administrative Access – All Covered Systems and Devices

Administrative access to a server must follow the user access requirements listed above. In addition:

1. Unnecessary administrative accounts *must* be removed or disabled.
2. Default passwords *must* be changed.
3. Administrative interfaces *must* never be left open and unattended. Use of locking screen savers and other methods *must* be used, even when the server is in a secure room.
4. Management interfaces *must* never be directly exposed to the Internet. Use of VPN or some other protective method *must* be used for off-campus access to these interfaces.
5. Servers should only be accessed over the network from “known secure” devices. Loss of a single administrator password to a keylogger can have devastating effects on a system.
6. Passwords for administrative accounts *must* meet or exceed the complexity requirements used within the provisioned “NetID” accounts. Where passwords longer than 8 characters are possible, try using 10 or more and rely on mnemonics and phrases.
7. Many administrative systems use shared passwords for specific functions. Such passwords *must* be changed whenever someone knowing the password leaves the associated administrative role or the password is compromised.
8. Increase password requirements as they give access to more sensitive data and systems. Never use a NetID password for accessing administrative functions.
9. Embedded cleartext passwords should be avoided. When this is necessary, careful use of access controls must be applied to prevent widespread access to the password. Such passwords need to be changed in the same way as any administrative password: whenever someone who knows the password changes roles, the password *must* be changed.

Exceptions

When the above standard cannot be met, the Unit IT Leader can either execute or request an exception. For situations involving a service limited to the unit or where the accounts are used for management of the system, the IT Leader can make exceptions based on their own judgment, but should attempt to stay as close to the standard as possible. For services that are provided to the entire campus, or large portions thereof, an exception request must be made.

The following lists some specific situations where an exception may be required:

1. In some cases, a small group of users will work with an application or system where they must share their access, and therefore will share the credentials. This creates a risk of no accountability, or of blaming someone else for one's own actions. When such situations must be supported, the owner of the system must make the situation known to the Unit Head, understand the risks, work towards removing such sharing of credentials, and make sure that no University Policies or state or federal regulations are violated by using such practices.
2. The service may simply not support User authentication. In this case, other protective methods must be employed to prevent unauthorized access to the service.

Exception Process

The Information Security Council and IT Security Officer provide an exception process for all situations where a unit or individual not in compliance with a University IT Security Policy, Standard, or Procedure may request an exception. This process can be found at [IT Security Exception Process](#).

Appendix A – Password Strength

How strong is strong enough? For “in band” (brute force guessing) attacks, there are government guidelines that require varying levels of password strength (and requirements for the overall authentication environment, such as defense against man-in-the-middle attacks) according to the sensitivity of the application where the password is used. For example, Level 1 assurance is a minimal requirement that is typically related to PIN-based security and only requires the password be able to resist brute force guessing such that the odds of a successful guess are no more than 1 in 2000 over the password's lifetime. A more relevant choice is Assurance Level 2, which is typical of the campus applications, and this defines acceptably strong passwords as passwords that have no more than a 1 in 65535 chance of being broken over the lifetime of the password. (Note that the lifetime factor is a key motivator for regular password changes.) Using this probability of 1 in 65535 as a baseline, let's examine what strength really means.

Strength is then really a function of the “entropy” of the password (how much randomness it may have) and how we regulate/throttle brute-force guessing via login attempts. Entropy can roughly be defined as the uncertainty an attacker has in his knowledge of the password: how hard it is to guess. Random passwords will have the maximum entropy and be the strongest. It's also easy to compute entropy of random passwords and we calculate entropy in bits. The number of bits of entropy per character in the password equals the log (base 2) of the size of the character set. If we take a large character set including all 94 possible keyboard characters and select l characters, we can have about 6.55 bits of entropy per character in the password, i.e., $l \times 6.55$. For the 79 character set used by the NetID password, this drops to about 6.25 bits per character. So an 8 character, random password using this character set has about 50 bits of entropy,

which is quite good. But in the real world, people choose their own (non-random) passwords and this makes estimation of the entropy (strength) more difficult. If you allow someone to choose a password with out any complexity requirements, entropy will be much lower than this 6.25 bits/character, and a generous estimate is that it will be 4-5 bits/character. For an 8 character password, you end up with only 32-40 bits of entropy. 32 bits of entropy for an 8 character password is pretty poor (see why below), so we need to force the users to add some complexity (variability) to the password and to avoid easily attacked dictionary words. It is estimated that these additional requirements increase the entropy by about 12 bits. If this is added to our lower estimate of 32 bits of entropy, we improve to 44 bits.

Now let's convert that to something useful: if we say that the password must be resistant to guessing at a rate of 1 in 65535, then the number of attempts we must limit the attacker to is 2^{44} divided by $2^{16} = 2.7$ million guesses. This means, using our rate limiting that constrains the attacker to 500,000 attempts per year, they will need 5 years to succeed. That's quite good. Why was the figure of 32 bits not so good? Divide 2^{32} by $2^{16} = 65536$ guesses, which means they need about 1.5 months. That's low enough to be of concern.

This is why we insist on the NetID rules listed above. If we could require passwords to be 10 or more characters, we could ease the complexity rules and allow alphanumeric passwords; this may be a future direction.

For more details on the above calculations, see <http://www.csrc.nist.gov/pki/twg/y2004/Presentations/twg-04-04.pdf> and articles by Jesper Johansson. Keep in mind we are estimating some of the values, such as the entropy, and it's possible the value for bits of entropy per character is even lower than 4. This points us towards the value of a yearly password change.

Keep in mind that all of this applies to brute-force attacks made against a login interface. An attacker who can obtain the password hashes (and crack them) or install a keylogger will have a much easier time of it, and even a complex 8-character password cannot resist those attacks.

1

See: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information By George A. Miller