Chapter 11 AUTHENTICATION

11.1 Authentication Basics

- **Subjects act on behalf of an external entity.**
  - **An example of a subject is a process running on a computer.**
  - **The identity of that entity controls the actions that its associated subjects may take.**
  - **Therefore, the subjects must bind to the identity of the external entity.**
- **Authentication is the binding of an identity to a subject.**
- **The external entity must provide information to enable the system to confirm its identity.**
  - **The information may come from one or more of the following:**
    - **What the entity knows (e.g. password)**
    - **What the entity has (e.g. badge, smart card)**
    - **What the entity is (e.g. fingerprints, retinal characteristics)**
    - **Where the entity is (e.g. in front of a particular terminal)**
- **The authentication process is consisted of the following steps:**
  - **Obtain the authentication information from an entity**
  - **Analyze the data**
  - **Determine if the data is associated with the entity**
- **The authentication process implies two things: the computer must store some information about the entity and the mechanisms to manage the data.**

Chapter 11 AUTHENTICATION

- **An authentication system is consisted of 5 components:**
  - **Set *A*** of authentication information: a set of specific information with which the entities prove their identities.
  - **Set *C*** of complementary information: a set of information that the system stores to validate the authentication information.
  - **Set *F*** of complementation functions: these functions generate the complementary information from the authentication information.

    For $f \in F$, $f : A \rightarrow C$.
  - **Set *L*** of authentication functions: these functions verify identity.

    For $l \in L$, $l : A \times C \rightarrow \{\text{true}, \text{false}\}$.
  - **Set *S*** of selection functions: These functions enable an entity to create or alter the authentication and complementary information.
- **EXAMPLE:**
  - A user authenticates himself by entering a password.
  - The system compares it with the cleartext password stored online.
  - *A* is the set of strings making up acceptable passwords.
  - $C = A$, $F = \{I\}$, and $L = \{eq\}$, where I is the identity function, and eq is true if its arguments are the same and false if they are not.

Chapter 11 AUTHENTICATION

11.2 Passwords

- A <u>password</u> is information associated with an entity that confirms the entity's identity.
- The <u>simplest</u> password is a sequence of characters.
- The password <u>space</u> is the set of all sequences of characters that can be used as passwords.
- EXAMPLE:
  - An installation requires each user to choose a sequence of 10 digits as a password.
  - The set *A* has 10^{10} elements (from "0000000000" to "9999999999").
- The set *C* may contain <u>more</u> or <u>fewer</u> elements than the set *A*, depending on the nature of the complementation function.
- The passwords may be stored in <u>protected</u> files or <u>in-the-clear</u>.
  - If the protected file is enciphered, the encipherment and decipherment keys are in the memory.
  - If the password file is compromised, all the passwords will be revealed.
- A good <u>solution</u> is to use a one-way function to hash the password into a complement.
  - If the password file is read, the attacker must do two things:
    - guess the passwords
    - invert the hash

Chapter 11 AUTHENTICATION

- **EXAMPLE: UNIX system**
  - **The original UNIX password mechanism does not store passwords in-the-clear.**
  - **Instead, it hashes each password into an 11-character string using one of 4096 hash functions.**
  - **Two characters that identify the hash function used are prepended.**
  - **The 13-character string is stored in a file.**
  - **A UNIX password is composed of up to 8 characters.**
  - **ASCII NUL (0) is not used.**
  - ***A* = { strings of 8 characters or less }, each character is chosen from a set of 127 possible characters.**
  - ***A* contains approximately $6.9\text{x}10^{16}$ passwords.**
  - ***C* = { 2-character hash id || 11-character hash }, each character chosen from an alphabet of 64 characters.**
  - ***C* contains approximately $3.0\text{x}10^{23}$ strings.**
  - ***F* = { 4096 versions of modified DES }, where the modification is based based upon a permutation of the DES.**
  - ***L* = { *login*, *su*, … }.**
  - ***S* = { *passwd*, *nispasswd*, *passwd+*, … }.**

Chapter 11 AUTHENTICATION

- **The goal of an authentication system is to ensure that entities are correctly identified.**
- **If one entity can guess another's password, the guesser can impersonate the other.**
- **The authentication model provides a systematic way to analyze this problem.**
  - **The goal is to find an $a \in A$ s.t. for $f \in F$, $f(a) = c \in C$.**
  - **c is associated with an entity.**
- **There are two approaches to determine whether *a* meets these requirements:**
  - **Hide enough information so that one of *a*, *c*, or *f* cannot be found.**
    - **EXAMPLE:**
      - **Many UNIX systems make the files containing the complementary information readable only by root.**
      - **These schemes use shadow password files, and make the elements in set *C* unknown.**
      - **Similarly, other systems make the elements in set *F* unknown.**
  - **Prevent access to the authentication functions *L*.**
    - **EXAMPLE:**
      - **One site does not allow the *root* user to login to an account from a network.**
      - **The login functions exist but will always fail.**
      - **Hence, one cannot test authentication of *root* with access to these functions over a network.**

Chapter 11 AUTHENTICATION

<u>Attacking a Password System</u>

- **The simplest attack against a password-based system is to guess passwords.**
- **A <u>dictionary attack</u> is the guessing of a password by repeated trial and error.**
  - **The name comes from the list of words used for guesses.**
  - **The dictionary may be a set of strings in <u>random order</u> or a set of strings in <u>decreasing order</u> of probability of selection.**
- **Dictionary attack <u>type 1</u>**
  - **If the complementary information or the complementation functions are available, the dictionary attack takes each guess g and computes $f(g)$ for each $f \in F$.**
  - **If $f(g)$ corresponds to the complementary information for entity E, g authenticates E under f.**
- **Dictionary attack <u>type 2</u>**
  - **If the complementary information or the complementation functions are unavailable, the authentication functions $l \in L$ may be used.**
  - **If the guess g results in l returning true, g is the correct password.**
- **EXAMPLE**
  - **<u>Type 1 attack</u>**
    - **Attackers often obtain a UNIX system's password file and use the (known) complementation function to test guesses.**
  - **<u>Type 2 attack</u>**
    - **The attackers need access to the system to obtain the complementary information in the password file.**
    - **To gain access, they may try to guess a password using the authentication function.**
    - **They use a known account name (such as *root*) and guess possible passwords by trying to login.**

Chapter 11 AUTHENTICATION

Countering Password Guessing

- **Password guessing requires either of two things:**
  - **The set of complementation functions and complementary information**
  - **Access to the authentication function**
- ***P*: probability of guessing a password in specified period of time.**
- ***G*: number of guesses tested in 1 time unit.**
- ***T*: number of time units during which guessing occurs.**
- ***N*: number of possible passwords**
- **Therefore $P \geq TG/N$**
- **EXAMPLE:**
  - ***R*: number of bytes per minute that can be sent over a communication line.**
  - ***E*: number of characters exchanged when logging in.**
  - **S: length of the password.**
  - ***A*: number of characters in the alphabet from which the characters of the password are drawn.**
  - **The number of possible passwords is $N=A^S$.**
  - **The number of guesses per minute is $G=R/E$.**
  - ***M*: number of months for password guessing.**
  - **This implies $T=4.32\text{x}10^4 M$.**

$$P \geq \frac{4.32x10^4 M(\frac{R}{E})}{A^S} \quad \Rightarrow \quad A^S \geq \frac{4.32x10^4 MR}{PE}$$

Chapter 11 AUTHENTICATION

- **EXAMPLE:**
  - **Passwords drawn from a 96-character alphabet**
  - **10^4 guesses can be tested per second**
  - **Probability of a successful guess is 0.5 over a 365 day period**
  - **What is minimum password length to give us this probability?**

$$N \geq \frac{TG}{P} = \frac{(365x24x60x60)10^4}{0.5} = 6.31x10^{11}$$

  - **Choose an integer *S* such that $\sum_{i=0}^{S} 96^i \geq N$ = 6.31x10^{11}.**
  - **The inequality holds when $S \geq 6$.**
  - **Hence, passwords must be at least 6 characters long.**

Chapter 11 AUTHENTICATION

Password Aging

- Guessing of passwords requires access to the complementary information, the complementation functions, and the authentication functions.
- If none of these have changed by the time the password is guessed, the attacker can use the password to access the system.
- Password aging is the requirement that a password be changed after some period of time has passed or after some event has occurred.
- In practice, aging by itself does not achieve much.
  - It should be used with other mechanisms discussed in this chapter.
- Two problems in implementing password aging
  - Force the users not to re-use passwords
  - Provide notice of the need to change and a user-friendly method of changing passwords
- Password aging is useless if a user changes the current password to the same thing.
  - This could be prevented if the *n* previous passwords are recorded by the system.
  - When the user changes the password, it is compared with the *n* passwords.
  - If there is a match, the new password is rejected.
- An alternative approached is based on time.
  - The user must change the password to one other than the current password.
  - The password cannot be changed for a minimum period of time.
  - This prevents the rapid cycling of passwords.
  - Unfortunately, it also prevents the user from changing the password if it is compromised within that time period.

Chapter 11 AUTHENTICATION

11.3 Challenge-Response

- **There is a fundamental problem with passwords: they are reusable!**
  - **If an attacker sees a password, she can later replay the password.**
  - **The system cannot distinguish between the attacker and the legitimate user.**
  - **An alternative is to authenticate the user in such a way that the transmitted password changes each time.**
- **U: user, S: system**
  - **U wants to authenticate himself to S.**
  - **U and S have agreed on a secret function f.**
  - **A challenge-response authentication system**
    - **S sends a random message m to U.**
    - **U replies with the response $r = f(m)$.**
    - **S validates r by computing it separately.**

<u>Pass Algorithms</u>

- **In a challenge-response authentication system, f is the <u>secret</u>. The function f is called a <u>pass algorithm</u>.**
  - **With this definition, no cryptographic keys or other secret information may be an argument for f.**
  - **The algorithm computing f is the secret.**

Chapter 11 AUTHENTICATION

One-time Passwords

- The ultimate form of password aging occurs when a password is valid for exactly one use.
- A one-time password is a password that is invalidated as soon as it is used.
- Problems associated with a one-time password scheme
  - Generation of random passwords: this problem can be solved by using a cryptographic hash function or an enciphering function such as DES.
  - Synchronization of the system and the user: a solution to this problem is to have a system that informs the user which password it expects (e.g., by having all the user's passwords numbered and the system providing the number of the one-time password it expects).
  - EXAMPLE: S/Key implements a one-time password scheme.
    - *h*: a one-way hash function (S/Key uses MD4 or MD5, depending on the version)
    - MD4 is a message digest algorithm by Professor Ronald Rivest of MIT in 1990.
    - MD5 was designed by Professor Ronald Rivest in 1991 to replace an earlier hash function, MD4. In 1996, a flaw was found with the design of MD5; while it was not a clearly fatal weakness, cryptographers began to recommend using other algorithms, such as SHA-1 (recent claims suggest that SHA-1 has been broken).
    - SHA-1, SHA-256, SHA-384, and SHA-512: other standards from NIST.
    - The user chooses an initial seed k.
    - The key generator calculates: $h(k)=k_1$, $h(k_1)=k_2, \ldots, h(k_{n-1})=k_n$.
    - The passwords, in the order they are used, are: $p_1=k_n$, $p_2=k_{n-1}, \ldots$, $p_{n-1}=k_1$.
    - Suppose an attacker intercepts p_i. The attacker needs to invert h, or launch a dictionary attack on h.
      - h is a one-way function that cannot be inverted.
      - For MD4 and MD5, dictionary attacks are not a threat if the seeds are chosen randomly.

Chapter 11 AUTHENTICATION

Hardware-Supported Challenge-Response Procedures

- Hardware support comes in 2 forms:
  - A program for a general purpose computer
  - Special-purpose hardware support
  - Both perform the same functions.
- The first type of hardware device provides mechanisms for hashing or enciphering information.
  - The system sends a challenge.
  - The user enters it into the device.
  - The device returns an appropriate response.
- The second type of hardware device is temporally based.
  - Every 60 seconds it displays a different number. The range of the numbers is: $[0,10^n-1]$.
  - A similar device is attached to the computer. It knows what number the device for each registered user should display.
  - To authenticate:
    - The user provides his login name.
    - The system requests a password.
    - The user enters the number shown on the hardware device, followed by a fixed (reusable) password.
    - The system validates that the number is the one expected for the user at that time and that the reusable portion of the password is correct.
  - EXAMPLE: The RSA SecureID card
    - Uses a system based on time.
    - The password is invalidated once a login succeeds.

Chapter 11 AUTHENTICATION

Challenge-Response and Dictionary Attacks

- **The attack is same as for fixed passwords.**
  - **If the attacker knows the challenge *r* and the response *f*(*r*), the dictionary attack proceeds.**
  - **In practice, it is not necessary to know the value of *r*.**
  - **Most challenges are composed of random data combined with public data that an attacker can determine.**
- **EXAMPLE:**
  - **A user is authenticating herself using a challenge-response system.**
  - **The system generates a random challenge *r*.**
  - **The user returns the value of $E_k(r)$.**
  - **The attacker knows both *r* and $E_k(r)$. He can try different values of *k* until the encipherment of *r* matches $E_k(r)$.**

Chapter 11 AUTHENTICATION

11.4 Biometrics

- **<u>Identification</u> by physical characteristics is as old as humanity.**
- **Biometrics is the <u>automated</u> measurement of biological, behavioral features that identify a person.**
- **When a user is given an account, the system administration takes a <u>set</u> of measurements that identify that user to an acceptable degree of error.**
- **The <u>biometric authentication mechanism</u> verifies the identity whenever the user accesses the system.**
- **A comparison to the known data for the claimed user's identity will either <u>verify</u> or <u>reject</u> the claim.**
- **Common characteristics are:**
  - **Fingerprints**
  - **Voice characteristics**
  - **Eyes**
  - **Facial features**
  - **Key stroke dynamics.**