Identification and Authentication

Why Authentication?

- Common policy requirement: restrict the behavior of a user
  To permit different users to do different things, we need a way to identify or distinguish between users
  - Identification mechanisms to indicate/provide identity
  - Authentication mechanisms to validate identity
- Authentication is a mutual process which may use different mechanisms (and therefore have different levels of assurance):
  - Users must prove their identity to the computer.
  - Computers must prove their identity to the user.
    (This also applies to processes and other computers … any subject wishing to interact)

Identification & Authentication

- When logging on to a computer you enter
  - user name and
  - password
- The first step is called identification:
  - You announce who you are.
- The second step is called authentication;
  - You prove that you are who you claim to be.
- To distinguish this type of 'authentication' from other interpretations, we refer here to user authentication: the process of verifying a claimed user identity.
- Authentication by password is widely accepted and not too difficult to implement.

Authentication System

- (A, C, F, L, S)
  - A set of authentication information used by entities to prove identity
  - C complementary information stored on computer and used by system to **validate** authentication information
  - F complementation functions $f: A \rightarrow C$ *to generate* $c=f(a)$
  - L functions that prove identity $l(a,c)$= T/F
  - S functions enabling entity to create or alter information in A or C

(bad) Example

- Password system, with passwords stored on line in clear text
  - *A* set of strings over fixed alphabet to construct the passwords
  - *C* = *A*
  - *F* singleton set of identity function { *I* }
  - *L* single equality test function { *eq* }
  - *S* functions to set/change password

User Authentication

Common mechanisms for "proving" user identity

- where the user is
  - access to the keyboard or IP address
- what the user knows
  - passwords, personal information
- what the user possesses
  - a physical key, a ticket, a passport, a token, a smart card, a badge
- what the user is (biometrics)
  - fingerprints, voiceprint, signature dynamics
- … or some combination of these

"Something you have"

- Very similar to the "something you know" technique - in order to implement it, there needs to be:
  - an object which may or may not be unique, but to which the access is limited to "authorized" users or other subjects
  - a way to present this object to the entity which requires the subject to provide proof
  - a way to determine if the object as presented is the one which was expected

Smart Cards

- A portable device with a CPU, I/O ports, and some nonvolatile memory (currently few thousand bytes) that is accessible only through its CPU
- It can carry out the computations required (for example by public key algorithms) and transmit results directly to the host
- Since devices are subject to theft, some devices require a PIN (something you know)
- PIN used by the device to authenticate the user
- Some use biometrics data about the user instead of the PIN

“Something you know”

- a word (password)
- an algorithm (pass-algorithm)
- a phrase (pass-phrase)
- a picture (pass-picture?)
- a combination or sequence of the above

- Authentication
  - Allows an entity (a user or a system) to prove its identity to another entity
  - Typically, the entity whose identity is verified *reveals knowledge* of some secret S to the verifier
  - Strong Authentication: The entity reveals knowledge of S to the verifier *without* revealing S to the verifier

Passwords

- Sequence of characters
  - Examples: 10 digits, a string of letters, *etc.*
  - Generated randomly, by user, by computer with user input
- Sequence of words
  - Examples: pass-phrases
- Algorithms
  - Examples: challenge-response, one-time passwords

How well does this work?

Ideal Policy: only a certain set of individuals are allowed into the system.

Stated Policy: only users having a valid password are allowed into the systems.

Actual Policy: permit users who

- Are issued a valid password (authenticator)
- Can obtain a valid authenticator
- Can bypass the authentication process

To get a valid authenticator…

- **Social engineering**
- **Guessing**: most break-ins occur because of bad passwords.
  - Do not use Your name (first, last, account name), Spouse, SO, pet, children, …, even with a single digit, Any word in any language, even with standard replacement (1=i, 0=o, ...)
- **Known/standard account** and password pairs
  Many systems have certain accounts set up with certain default passwords (either well known or easy to guess). UNIX provides the guest account, with password often GUEST! VAX/VMS used to come with FIELD/SERVICE.
- **Known algorithms** for assigning passwords
  - use some/part of SSN, birthday, name, student/employee id, account name, phone extension

Social Engineering

- Univ. of Sydney study (1996)
  - 336 CS students emailed asking for their passwords
    - Pretext: "validate" password database after suspected break-in
  - 138 returned their passwords; 30 returned invalid passwords; 200 reset passwords (not disjoint)
- Treasury Dept. report (2005)
  - Auditors pose as IT personnel attempting to correct a "network problem"
  - 35 (of 100) IRS managers and employees provide their usernames and change passwords to a known value

Problem: pswd Storage

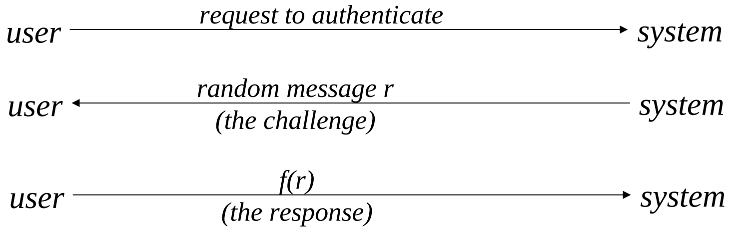
- Store as cleartext
  - If password file compromised, *all* passwords revealed
- Encipher file
  - Need to have decipherment, encipherment keys in memory
  - Reduces to previous problem (where is the key?)
- Store one-way hash of password
  - If the file is read, attacker must still guess passwords or invert the hash (but where is the hash?)

Example

- UNIX system standard hash function
  - Hashes password into 11 printable char string using one of 4096 hash functions
- As authentication system:
  - *A* = { strings of 8 chars or less }
  - *C* = { 2 char hash id || 11 char hash }
  - *F* = { 4096 versions of modified DES }
  - *L* = { *login*, *su*, … }
  - *S* = { *passwd*, *nispasswd*, *passwd+*, … }

Challenge-Response

User and system share a secret function *f* (in practice, *f* is a known function with unknown parameters, such as a cryptographic key)



Pass Algorithms

- Challenge-response with the function *f* itself a secret
  - Example:
    - Challenge is a random string of characters such as "abcdefg", "ageksido"
    - Response is some function of that string such as "bdf", "gkip"
  - Can alter algorithm based on ancillary information
    - Network connection is as above, dial-up might require "aceg", "aesd"
  - Usually used in conjunction with fixed, reusable password

What is the advantage over passwords?

- Avoids "replay" attacks
- One-time password
  - authentication information a changes after each use
  - Why is this challenge-response?

Attack

- Attacker knows (space of) encryption function
- Captures challenge and response
- Learns encryption function / key
- Can now properly respond to new challenge

Solution: encrypt challenge

- Use shared key to share session key
- Session key encrypts challenge
- Challenge thus indistinguishable from random data

Dictionary Attacks

- Trial-and-error from a list of potential passwords
  - *Off-line*: attacker knows A, *f* and *c*'s, and repeatedly tries different guesses $g \in A$ until the list is done or passwords guessed
    - Examples: *crack*, *john-the-ripper*
  - *On-line*: have access to functions in *L* and try guesses *g* until some *l* (*g*) succeeds
    - Examples: trying to log in by guessing a password

Using Time to counter guessing

Anderson's formula:

- *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
- *N* number of possible passwords ($|A|$)
- Then $P \geq TG/N$

Example

- Goal
  - Passwords drawn from a 96-char alphabet
  - Can test 10^4 guesses per second
  - Probability of a success to be 0.5 over a 365 day period
  - What is minimum password length?
- Solution
  - $N \geq TG/P = (365 \times 24 \times 60 \times 60) \times 10^4 / 0.5 = 6.31 \times 10^{11}$
  - Choose s such that $\Sigma^{s}_{j=0} 96^j \geq N$
  - So $s \geq 6$, meaning passwords must be at least 6 chars long

First UNIX Password Scheme

- [Wilkes68] (recall DES was 1976)
- Encryption based on M-209 cipher machine (US Army WWII)
- Easy to invert unknown plaintext and known key, used password as key:
  - Instead of E_K (password) used hash function $E_{Password}$ (0)
- PDP-11 could check all 5 or less letter lower-case passwords in 4 hours!

Making Brute Force Attacks Harder

- Use a slower encryption (hashing) algorithm
  - Switched to DES: $H(p) = \mathrm{DES}_p(0)$
- Even slower: run DES lots of times
  - UNIX uses $\mathrm{DES}_p^{25}(0)$

    $\ldots \mathrm{DES}_p(\mathrm{DES}_p(\mathrm{DES}_p(\mathrm{DES}_p(0))))$
- Require longer passwords
  - DES key is only 56 bits: only uses first 7.5 characters (ASCII)
  - 95 printable characters, $95^8 = 6.6 * 10^{15}$

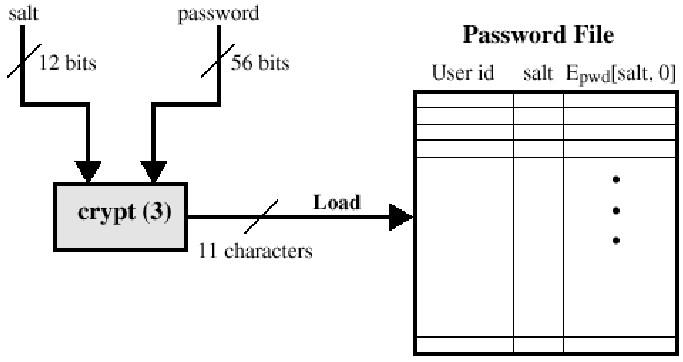
UNIX Passwords

- UNIX passwords were kept in a publicly readable file, etc/passwords.
- Now they are often kept in a "shadow" directory and only visible by "root".
- The salt serves three purposes:
  - Prevents duplicate passwords.
  - Effectively increases the length of the password.
  - Prevents the use of hardware implementations of DES

Password Salt

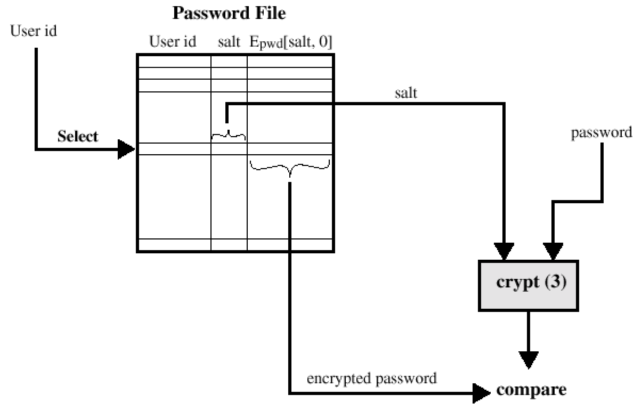
- It is used to make the dictionary attack a bit more difficult
- Salt is a 12 bit number between 0 and 4095
- It is derived from the system clock and the process identifier
- Rather than computing F(pwd), F(pwd + salt) is computed; both salt and F(pwd + salt) are stored in the password table
- When a user supplies the password, system fetches the salt for the user and computes F(pwd + salt) to check for a match
- Notice that with salt, the same password is computed in 4096 different ways

UNIX Password Scheme



Loading a new password

UNIX Password Scheme



Verifying a password file

Dictionary Attacks on Passwords

- Attack 1:
  - Create a dictionary of common words and names and their simple transformations and use them to guess the password
- Attack 2:
  - Usually F is public and so is the password file
    - In Unix, F is crypt(3) and /etc/passwd may be world readable
  - Compute F(word) for each word in the dictionary
  - A match gives the password
- Attack 3:
  - To speed up search, pre-compute F(dictionary)
  - A simple look up gives the password
- Note that these attacks work only with weak passwords

Password Management Policy and Procedure

- Educate users to make better choices
  - Does not work if the user population is large or novice
- Define rules for good password selection and ask users to follow them
  - Rules serve as guideline for attackers
- Ask or force users to change their passwords periodically
- Force users to use machine generated passwords
  - Random password are difficult to memorize; also password generator may become known to the attacker through analysis
- Actively attempt to break users' passwords; force users to change those that are broken
  - Attacker may have better dictionary
- Screen password choices; if a choice is weak, force users to make a different choice

Single Sign-on

- Having to remember many passwords for different services is a nuisance; with a single sign-on service, you have to enter your password only once.
- A simplistic single-sign on service could store your password and do the job for you whenever you have to authenticate yourself.
  - Such a service adds to your convenience but it also raises new security concerns.
- System designers have to balance convenience and security; ease-of-use is an important factor in making IT systems really useful, but many practices which are convenient also introduce new vulnerabilities.

One-Time Passwords

- Password that can be used exactly *once*
  - After use, it is immediately invalidated
- Challenge-response mechanism
  - Challenge is number of authentications; response is password for that particular number
- Problems
  - Synchronization of user, system
  - Generation of good random passwords
  - Password distribution problem

Lamport's Scheme

- Does not require any special hardware
- User selects x and computes F(x), $F^2(x)$, ..., $F^{100}(x)$ (This will allow 100 logins before a seed change)
- System stores (User name, $F^{100}(x)$) (need not know x)
- User supplies y = $F^{99}(x)$ the first time
- System computes F(y) and compares it with $F^{100}(x)$
- If they match, the login is correct and the system replaces $F^{100}(x)$ by $F^{99}(x)$
- User supplies $F^{98}(x)$ the next time, and so on
- Knowing (intercepting) y does not reveal the next password ($F^{-1}(y)$) if F is a one-way function
- User calculates $F^n(x)$ using a hand-held calculator, a trusted workstation, or a portable computer
- In Bellcore's implementation of this scheme, called S/Key, user calculates the sequence on a secure machine, encodes it as a sequence of short words, and prints it

S/Key

- One-time password scheme based on idea of Lamport
- *h* one-way hash function (MD5 or SHA-1, for example)
- User chooses initial seed *k*
- System calculates:

$$h(k) = k_1,\ h(k_1) = k_2,\ \ldots,\ h(k_{n-1}) = k_n$$

- Passwords are reverse order:

$$p_1 = k_n,\ p_2 = k_{n-1},\ \ldots,\ p_{n-1} = k_2,\ p_n = k_1$$

S/Key Protocol

System stores maximum number of authentications n, number of next authentication i, last correctly supplied password p_{i-1}.

user —{ *name* }→ *system*

user ←{ *i* }— *system*

user —{ p_i }→ *system*

System computes $h(p_i) = h(k_{n-i+1}) = k_{n-i} = p_{i-1}$. If match with what is stored, system replaces p_{i-1} with p_i and increments i.

"Something about you"

- Biometrics are increasingly common as identification rates improve.
  - fingerprints
  - retinal scan, iris scan
  - facial heat
  - voice pattern/recognition
  - signatures (handwriting)
  - typing
- See also:
  - U.S. National Biometric Test Center; San Jose State Univ. (CA)
  - www.nist.gov/biometrics

Some typical biometrics

- Primarily Physical Features
  - Hand based
    - Fingerprint or fingerscan
    - Hand geometry
  - Face/eye
    - Facial recognition
    - Retinal scans / Iris scans
- Strong Behavioral Component
  - Voice recognition
  - Signature recognition, including how the signature is produced (pressure, speed, stroke order) and not just how the signature looks
  - Typing style, including speed and rhythm of key pressure

How does this work?

- Some aspects are quite similar to standard authentication procedures
  - Calibrate and store user information
    - Storage styles vary:
      - Common way in '99 was to encrypt user biometric information and store it
      - Alternate method would be to store a validator for the biometric information (hash, Unix-style validator)
  - Authenticate "as usual"
    - User "inputs" biometric info
      - (this might not be overt, and might not be a single event)
    - Proceed as with password techniques.

Matches are probabilities

- Identifying information is not typed in, but obtained by a device (imprecise measurement)
  - Characteristics mapped from analog to digital and not all of the original information is retained
  - Devices for most common biometrics may not produce identical results or even identically repeatable results
    - Ex: fingerprint readers depend on environmental factors such as the positioning of the finger, the "moisture" of the hand, oils, and occupational issues which may cause a print to be roughened over time

Effectiveness

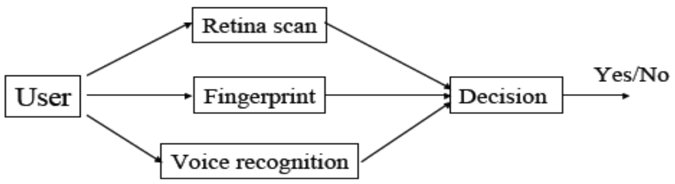
- Two types of errors for authentication
  - False acceptance (FA)
    - Let imposters in
    - FAR: the probability that an imposter is authenticated.
  - False rejection (FR)
    - Keep authorized users out
    - FRR: the probability that an authorized user is rejected.
- Another type of error for identification
  - False match (FM)
    - One user is mistaken for another (legitimate user)
    - FMR: the probability that a user is incorrectly matched to a different user's profile.
- No technique is perfect!

Other Characteristics

- Can use several other characteristics
  - Eyes: patterns in irises unique
    - Measure patterns, determine if differences are random; or correlate images using statistical tests
  - Faces: image, or specific characteristics like distance from nose to chin
    - Lighting, view of face, other noise can hinder this
  - Keystroke dynamics: believed to be unique
    - Keystroke intervals, pressure, duration of stroke, where key is struck
    - Statistical tests used

Multimodal Biometrics

- Use multiple Biometrics together.
  - AND: Accept only when all are passed
    - Why do we need this?
  - OR: Accept as long as at least one is passed
    - Why do we need this?
  - Others



Cautions

- These can be fooled!
  - Assumes biometric device accurate *in the environment it is being used in!*
  - Transmission of data to validator is tamperproof, correct

Higher Storage Requirements

- "Size" of the template as stored can be quite large in comparison with a password and is not necessarily directly tied to the accuracy
- Some typical template sizes:
  - Fingerscan: 250 - 100 bytes
  - Hand geometry: 9-20 bytes
  - Iris: 512 bytes
  - Retina: 96 bytes

Devices Usually Required

- The device collecting the data is often proprietary and/or uses proprietary algorithms
- Patents protect much of the technology
- There may be considerable computation involved in computing a "validator" or template for storage (far beyond the Unix validator)
- Sometimes the biometric requires local installation of a specialized reader device (such as for fingerprints, but not for voice)

Costs

Technique	Description	Min. Cost	False Reading
Retina	Eyes scanned 1 to 2 inches from screening device	$2,400	1/10,000,000+
Iris	Camera image of eye takes from 14 inches	$3,500	1/131,000
Hand	Hand scanned on plate by three video cameras at different angles	$2,150	1/500
Fingerprint	Finger scanned on glass plate	$1,995	1/500
Signature	Written with special pen on digitizer tablet	$1,000	1/50
Voice	Predefined phrase spoken into telephone or microphone	$1,500	1/50

Identity

Authentication is the binding of an identity to a subject
But what is identity?
A set of properties/attributes characteristic of a principal (subject or object)

How to represent identity?

- randomly chosen : not useful to humans
- user chosen: probably not unique globally
- hierarchical system: used to disambiguate
    - file system
    - X.500
    - IP address

To verify identity

Authentication: does subject match identity?

- Problem: does identity match principal ?
- Solution: certificate
  - validation that identity belongs to known principal
  - Certification Authority issues certificate user chosen: probably not unique globally
  - CA is trusted

Certificate Examples

Verisign

- Independently verifies identity of principal
- Levels of certification
  - Email address verified
  - Name/address verified
  - Legal identity verified
- More common: *corporate* identity
  - Is this really PayTuition.EDU I am giving my bank account number to?

PGP (Pretty Good Privacy): “Web of Trust”

- Users verify/sign certificates of other users
- Do I trust the signer?
  - or someone who signed their certificate?

Anonymity

- What if identity not needed?
  - Web browsing
  - Complaints about assignments
- Removing identity not as easy as it sounds
  - I can send email without my userid
  - But it still traces back to my machine
- Solution: anonymizer
  - Strips identity from message
  - Replaces with (generated) id
  - Send to original destination
  - Response: map generated id back to original identity

Anonymity

- Problem: Anonymizer knows identity
  - Can it be trusted?
  - Courts say no!
- Solution: multiple anonymizers
  - Onion Routing
  - Crowds