IS 2150 / TEL 2810
Information Security & Privacy

James Joshi
Associate Professor, SIS

Lecture 8
Feb 24, 2015

Authentication, Identity

Objectives

- Understand/explain the issues related to, and utilize the techniques
  - Authentication and identification
  - Passwords

Authentication and Identity

What is Authentication?

- Authentication:
  - Binding identity and external entity to subject
- How do we do it?
  - Entity *knows* something (secret)
    - Passwords, id numbers
  - Entity *has* something
    - Badge, smart card
  - Entity *is* something
    - Biometrics: fingerprints or retinal characteristics
  - Entity is in *someplace*
    - Source IP, restricted area terminal

Authentication System: Definition

- *A*: Set of *authentication information*
  - used by entities to prove their identities (e.g., password)
- *C*: Set of *complementary information*
  - used by system to validate authentication information (e.g., hash of a password or the password itself)
- *F*: Set of *complementation functions* (to generate *C*)
  - $f : A \rightarrow C$
  - Generate appropriate $c \in C$ given $a \in A$
- *L*: set of *authentication functions*
  - $l: A \times C \rightarrow$ { **true, false** }
  - verify identity
- *S*: set of *selection functions*
  - Generate/alter *A* and *C*
  - e.g., commands to change password

Authentication System: Passwords

- Example: plaintext passwords
  - $A = C =$ alphabet*
  - f returns argument: $f(a)$ returns a
  - l is string equivalence: $l(a, b)$ is true if $a = b$

- Complementation Function
  - Null (return the argument as above)
    - requires that c be protected; i.e. password file needs to be protected
  - One-way hash – function such that
    - *Complementary information* $c = f(a)$ easy to compute
    - $f^{-1}(c)$ difficult to compute

Passwords

- Example: Original Unix
  - A password is up to eight characters
    - each character could be one of 127 possible characters;
  - *A* contains approx. 6.9×10^{16} passwords
  - Password is hashed using one of 4096 functions into a 11 character string
  - 2 characters pre-pended to indicate the hash function used
  - *C* contains passwords of size 13 characters, each character from an alphabet of 64 characters
    - Approximately 3.0×10^{23} strings
  - Stored in file */etc/passwd* (all can read)

Authentication System

- Goal: identify the entities correctly
- Approaches to protecting
  - Hide enough information so that one of *a, c* or *f* cannot be found
    - Make C readable only to root
    - Make F unknown
  - Prevent access to the authentication functions *L*
    - *root* cannot log in over the network

Attacks on Passwords

- Dictionary attack: Trial and error guessing
  - Type 1: attacker knows *A*, *F*, *C*
    - Guess *g* and compute *f*(*g*) for each *f* in *F*
  - Type 2: attacker knows *A*, *I*
    - *I* returns **True** for guess *g*

- Counter: Difficulty based on |*A*|, Time
    - Probability *P* of breaking a password
    - *G* be the number of guesses that can be tested in one time unit
    - $|A| \geq TG/P$
    - Assumptions:
      - time constant; all passwords are equally likely

Password Selection

- Random
  - Depends on the quality of random number generator;
  - Size of legal passwords
    - 8 characters: humans can remember only one

- Pronounceable nonsense
  - Based on unit of sound (phoneme)
  - Easier to remember

- User selection (proactive selection)
  - Controls on allowable
    - At least 1 digit, 1 letter, 1 punctuation, 1 control character
    - Obscure poem verse

Password Selection

- Reusable Passwords susceptible to dictionary attack (type 1)
  - *Salting* can be used to increase effort needed
    - makes the choice of complementation function a function of randomly selected data
    - Random data is different for different user
    - Authentication function is chosen on the basis of the salt
    - Many Unix systems:
      - A salt is randomly chosen from 0..4095
      - Complementation function depends on the salt

Password Selection

- Password aging
  - Change password after some time: based on expected time to guess a password
  - Disallow change to previous *n* passwords
- Fundamental problem is *reusability*
  - Replay attack is easy
  - Solution:
    - Authenticate in such a way that the transmitted password changes each time

Authentication Systems: Challenge-Response

- Pass algorithm
  - authenticator sends message m
  - subject responds with $f(m)$
    - f is a secret encryption function
  - Example: ask for second input based on some algorithm

Authentication Systems: Challenge-Response

- One-time password: *invalidated after use*
  - f changes after use
- S/Key uses a hash function (MD4/MD5)
  - User chooses an initial seed k
  - Key generator calculates
    - $k_1 = h(k)$, $k_2 = h(k_1)$ …, $k_n = h(k_{n-1})$
  - Passwords used in the order
    - $p_1 = k_n$, $p_2 = k_{n-1}$, …, $p_n = k_1$
  - Suppose $p_1 = k_n$ is intercepted;
    - the next password is $p_2 = k_{n-1}$
    - Since $h(k_{n-1}) = k_n$, the attacker needs to invert h to determine the next password

Authentication Systems: Biometrics

- Used for human subject identification based on physical characteristics that are tough to copy
  - Fingerprint (optical scanning)
    - Camera's needed (bulky)
  - Voice
    - Speaker-verification (identity) or speaker-recognition (info content)
  - Iris/retina patterns (unique for each person)
    - Laser beaming is intrusive
  - Face recognition
    - Facial features can make this difficult
  - Keystroke interval/timing/pressure

Attacks on Biometrics

- Fake biometrics
  - fingerprint “mask”
  - copy keystroke pattern
- Fake the interaction between device and system
  - Replay attack
  - Requires careful design of entire authentication system