Computer Systems Security

Lesson 02 – Authentication
(Username and password)

Username and passwords

- Authentication controls
  - Password length
  - Password complexity or filters
  - Password history
  - Maximum password age
  - Minimum password age
  - Account expiration
  - Account restrictions
  - Account lockout

Usernames and Passwords

- Password authentication scheme is a logical combination of:
  - Input characters
  - Algorithms
    - Generating characters corresponding to input code
    - Authentication
  - Legitimate users password database
  - Form interface

Ideal Password Authentication scheme

- Should achieve the following goals:
  - The passwords or verification tables are not stored in the system
  - The passwords can be chosen and changed freely by the users.
  - The passwords cannot be revealed by the administrator of the server.
  - The passwords are not transmitted in plain text over the network
  - The length of a password must be appropriate for memorization.

Ideal Password Authentication scheme

- Should achieve the following goals:
  - The scheme must be efficient and practical
  - Any unauthorized login can be quickly detected when a user inputs a wrong password.
  - A session key is established during the password authentication process to provide confidentiality of communication.
  - The ID should be dynamically changed for each login session to avoid partial information leakage about the user's login message.

Password Authentication Schemes

- Common Password Authentication schemes:
  - RSA Scheme
  - ElGamal Scheme
  - Hash based
- To be discussed in details in Cryptography

RSA Password Scheme

- Based on the RSA public key cryptosystem
  - Concept: (more later)
    - Two large prime numbers: p,q
    - Compute products: n =pxq
    - Compute totient of p and q
    - Choose two relatively prime numbers: e and d such that e x d mod totient p,q = 1
  - Do not store passwords or verification tables in the server
  - Lets the users freely exchange their passwords
  - Vulnerable to replay attacks

Password authentication schemes

- Security requirements:
  - Withstand Denial of Service attacks
  - Withstand impersonation (forgery)
  - Forward secrecy:
    - Ensures that the previously generated passwords in the system are secure even if the system's secret key has been revealed in public by accident or is stolen
  - Mutual authentication:
    - Both the server and the user can identify each other

Password authentication schemes - Security Requirements

- Security requirements:
  - Protect against Parallel session attacks
    - Guard against eavesdropping
  - Protect against Password guessing attacks
  - Protect against Replay attacks

Passwords storage

Filing System

Clear text

Dedicated Authentication Server

Clear text

Encrypted

Password + Encryption = bf4ee8HjaQkbw

Hashed

Password + Hash function = aad3b435b51404eeaad3b435b51404ee

Salted Hash

(Username + Salt + Password) + Hash function =
e3ed2cb1f5e0162199be16b12419c012

Passwords storage - Hashing

- Usually stored as hashes (not plain text)
  - Plain-text is converted into a message digest through use of a hashing algorithm (i.e. MD5, SHA)

Passwords storage - Hashing

- Hash function H must have some properties:
  - One-way: given H(password), hard to find password
    - No known algorithm better than trial and error
  - Collision-resistant: given H(password1), hard to find password2 such that: H(password1) = H(password2)
    - It should even be hard to find any pair p1,p2 s.t. H(p1)=H(p2)

Passwords storage – Early UNIX systems

- In past UNIX systems, password used modified DES (encryption algorithm) as if it were a hash function
  - Encrypts NULL string using password as the key (truncates passwords to 8 characters!)
  - Caused artificial slowdown: ran DES 25 times
- Also stored password file in directory: /etc/passwd/
  - World-readable (anyone who accessed the machine would be able to copy the password file to crack at their leisure)
  - Contained userIDs/groupIDs used by many system programs
  - Can instruct modern UNIXes to use MD5 hash function

Passwords storage – Newer UNIX systems

- Password hashes stored in /etc/shadow directory (or similar)
  - only readable by system administrator (root)
- Less sensitive information still in /etc/password
- Added expiration dates for passwords

- Early “shadow” implementations on Linux called the login program which had a buffer overflow!

Passwords storage – Windows NT/2k/XP/Vista and later

- Uses 2 functions for “hashing” passwords:
  1. LAN Manager hash (LM hash)
     - Password is padded with zeros until there are 14 characters.
     - It is then converted to uppercase and split into two 7-character pieces
     - Each half is encrypted using an 8-byte DES (data encryption standard) key
     - Result is combined into a 16-byte, one way hash value
  2. NT hash (NT hash)
     - Converts password to Unicode and uses MD4 hash algorithm to obtain a 16-byte value
- Hashes stored in Security Accounts Manager (SAM)
  - Locked within system kernel when system is running.
  - Location - C:\WINNT\SYSTEM32\CONFIG
- SYSKEY
  - Utility which moves the encryption key for the SAM database off of the computer

Attacking Passwords

- Authentication system requirements:
  - Authentication Information (set A):
    - Set of specific information with which entities prove their identities.
  - Complimentary information (set C):
    - Set of information the system stores and uses to validate the authentication information.
  - Complementation functions (set F):
    - Functions that generate complementary information from the authentication information. i.e. for $f \in F$, f: A $\leftrightarrow$C

Attacking Passwords

- Authentication system requirements (contd…):
  - Authentication functions (set L)
    - Functions that verify identity.
    - i.e. *l* ∈L: A x C →{**true, false**}
  - Selection functions (set S):
    - Functions that enable an entity to create or alter the authentication and complementary information.

Attacking a password system

- Dictionary Attack
- Bruteforce attack

Dictionary Attack

- Definition:
  - Guessing of a password by repeated trial and error
  - Dictionary:
    - Set of string in a random order
    - Set of strings in a decreasing order of probability of selection.
  - Requires either the set of complementation functions and complementary information or access to the authentication functions.

Dictionary Attack

- Type 1:
  - Relies on availability of complementary information and complementation functions
  - Takes a guess (*g*) from the list of strings,
  - Computes *f(g) for each f ∈ F.* If *f(g)* corresponds to the complementary information for an entity, E, then *g* authenticates E under *f*.

Dictionary Attack

- Type 2:
  - Either complementary information or the complementation functions are unavailable.
  - Authentication functions $l \in L$ may be used.
  - If the guess g results in l returning **true**, g is the correct password.