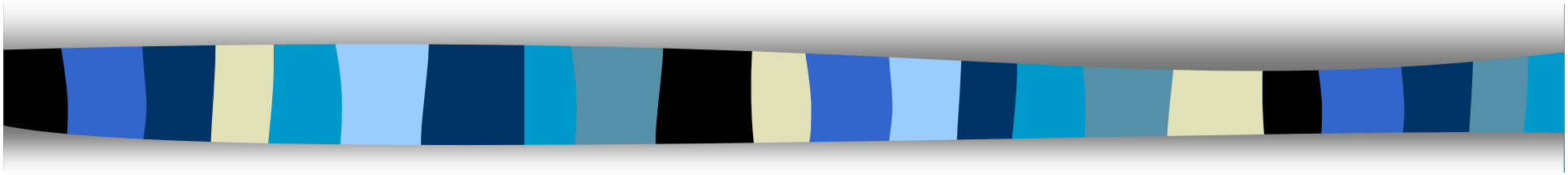
Computer Security
CS 426
Lecture 8



User Authentication

User Authentication

- Using a method to validate users who attempt to access a computer system or resources, to ensure they are authorized
- Types of user authentication
  - Something you know
    - User accounts with passwords
  - Something you have
    - Smart cards or other security tokens
  - Something you are
    - Biometrics

Scenarios Requiring Authentication

- Scenarios
  - Logging into a local computer
  - Logging into a computer remotely
  - Access web sites
- Potential vulnerabilities to consider when client authenticating server
  - channel between the client and the server
  - server compromise
  - client compromise
  - social engineering
  - weak passwords
- What are some threats?

Threats to Passwords

- Online guessing attempts
- Offline dictionary attacks
- Login spoofing
- Shoulder surfing
- Social engineering
  - e.g., pretexting: creating and using an invented scenario (the pretext) to persuade a target to release information or perform an action and is usually done over the telephone

Storing Passwords (UNIX Case Study)

- Old UNIX
  - The file /etc/passwd stores H(password) together with each user's login name, user id, home directory, login shell, etc.
  - file must be world readable
  - Brute force attacks possible even if H is one-way
    - how to brute-force when trying to obtain password of any account on a system with many accounts?
- New UNIX
  - H(password) stored in /etc/shadow, readable only by root

Password Salts

- Store [r, H(password,r)] rather than H(password)
  - r is randomly chosen for each password
  - r is public
  - similar to Initial Vector in CBC & CTR modes
- Benefits
  - dictionary attacks much more difficult
    - cost of attacking a single account remains the same
  - if two users happen to choose the same password, it doesn't immediately show

Mechanisms to Defend Against Dictionary and Guessing Attacks

- Protect stored passwords (use both cryptography & access control)
- Disable accounts with multiple failed attempts

Mechanisms to Avoid Weak Passwords

- Allow long passphrases
- Randomly generate passwords
- Check the quality of user-selected passwords
  - use a number of rules
  - run dictionary attack tools
- Give user suggestions/guidelines in choosing passwords
  - e.g., think of a sentence and select letters from it, “It’s 12 noon and I am hungry” => “I’S12&IAH”
  - Using both letter, numbers, and special characters
- Mandate password expiration
- Things to remember: Usability issues

Mechanisms to Defend Against Login Spoofing: Trusted Path

- Attacks:
  - write a program showing a login window on screen and record the passwords
  - put su in current directory
- Defense: Trusted Path
  - Mechanism that provides confidence that the user is communicating with what the real server (typically Trusted Computing Base in OSes)
    - attackers can't intercept or modify whatever information is being communicated.
    - defends attacks such as fake login programs
  - Example: Ctrl+Alt+Del for log in on Windows

Defending Against Other Threats

- Use ideas from recent research:
  - graphical passwords,
  - combine with typing
- Go beyond passwords
  - security tokens
  - biometrics
  - 2-factor authentication
    - US Banks are required to use 2-factor authentication by end of 2006 for online banking

Using Passwords Over Insecure Channel

- One-time passwords
  - Each password is used only once
  - Defend against passive adversaries who eavesdrop and later attempt to impersonate
- Challenge response
  - Send a response related to both the password and a challenge
- Zero knowledge proof of knowledge

How to do One-Time Password

- Time-synchronized OTP

- Shared lists of one-time passwords
- Using a hash chain (Lamport)
  - h(s), h(h(s), h(h(h(s))), …, h^{1000}(s)
  - use these values as passwords in reverse order

Lamport's One-Time Password: Details

- One-time setup:
  - A selects a value w, a hash function H(), and an integer t, computes $w_0 = H^t(w)$ and sends w_0 to B
  - B stores w_0
- Protocol: to identify to B for the i^{th} time, $1 \leq i \leq t$
  - A sends to B: A, i, $w_i = H^{t-i}(w)$
  - B checks $i = i_A$, $H(w_i) = w_{i-1}$
  - if both holds, $i_A = i_A + 1$

Challenge-Response Protocols

- Goal: one entity authenticates to other entity proving the knowledge of a secret, ‘challenge’
- Approach: Use time-variant parameters to prevent replay, interleaving attacks, provide uniqueness and timeliness
  - e.g., nounce (used only once), timestamps

Challenge-response based on symmetric-key crypto

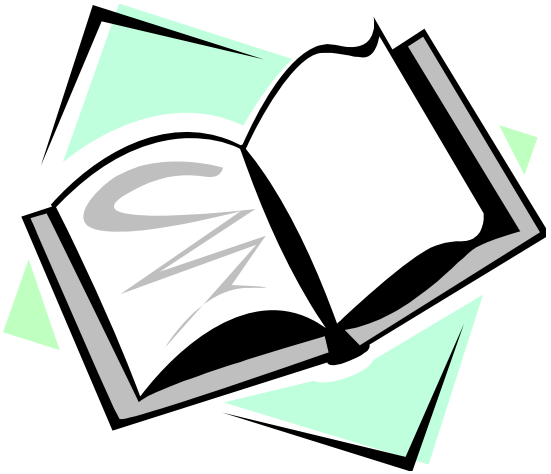
- Unilateral authentication, timestamp-based
  - A to B: $MAC_K(t_A, B)$
- Unilateral authentication, nounce-based
  - B to A: r_B
  - A to B: $MAC_K(r_B, B)$
- Mutual authentication, nounce-based
  - B to A: r_B
  - A to B: r_A, $MAC_K(r_A, r_B, B)$
  - B to A: $MAC_K(r_B, r_A)$

Issues to Consider in Password Systems

- Which types of attacks to defend against?
  - targeted attack on one account
  - attempt to penetrate any account on a system
  - attempt to penetrate any account on any system
  - service denial attack
- Whether to protect users against each other?
- Can users be trained? Will they follow the suggestions?
- Will the passwords be used in other systems?
- Whether the passwords will be used in a controlled environment?

Readings for This Lecture

- Wikipedia
  - Salt_(cryptography)
  - Password cracking
  - Trusted path
  - One time password



Coming Attractions …

- Unix Access Control