User Authentication Protocols

Week 4
User Authentication

- The process of verifying an identity claimed by a system entity
- Fundamental system security building block
  - Basis of access control & user accountability
- Has two steps:
  - Identification – provide claimed identity
  - Authentication – verify validity of claim
- User authentication ≠ message authentication
User Authentication: How?

- Based on something the individual knows:
  - Knows - e.g. password, PIN
  - Possesses - e.g. key, token, smartcard
- Is (static biometrics): fingerprint, retina
- Does (dynamic biometrics): voice, handwriting

- Can use alone or combined
- All can provide user authentication
- *All have issues*
Authentication Protocols

- Convince parties of each other's identity
  - Also exchange session keys
- May be one-way or two-way (mutual)

Key issues:

1. Confidentiality
   - Protect session keys
   - Prior keys or secrets need to exist
2. Timeliness
   - Prevent replay attacks
Replay Attacks

- Valid signed message is copied and later re-sent
- Simple replay
  - Copy message; replay later
- Repetition that can be logged
  - Replay timestamped message within validity interval
- Repetition that cannot be detected
  - Suppress original message
- Backward replay without modification
  - Send the replay message back to its sender
Replay Attacks: Countermeasures

- Sequence numbers
  - Attach sequence number *seqno* to message
  - Accept message if *seqno* follows previous value
  - Not always practical

- Timestamps
  - Message needs to contain *timestamp*
  - Accept message if timestamp is within validity window
  - Need synchronized clocks
Countermeasures (cont’d)

- Challenge/response
  - Ensures message *freshness*
  - Challenger sends random nonce R
  - Responder’s message needs contain a function of R
Authentication

- One-way authentication
- Mutual: two-way authentication
  - Using symmetric key crypto
  - Using public-key crypto
One-Way Authentication

Alice

Login A

Mallory

Trent T (Host)

How can T know it’s Alice and not Mallory impersonating Alice?
Authentication Approaches

- **Password**
  - Host stores Alice’s password
  - Alice sends password
  - Host verifies password

- **Problem:**
  - Trent stores all passwords in clear
  - Whoever breaks into Trent can steal passwords

- **Solutions**
  - One-Way Functions
  - Dictionary Attacks and Salts
Authentication Using Hashes

- Roger Needham and Mike Guy
  - T does not need to know password
  - Only differentiate between valid and invalid ones

Alice

Problem?

Login A, pwd

User ID

H(pwd)

T: Compare H(pwd) to $H_A$

Password file

Alice

$H_A$

Trent T (Host)
Password Vulnerabilities

- One-way hashes are vulnerable
  - *Which password is better?*
    - Barney
    - 9(hH/A.
- Which one is easier to remember?
- Dictionary attack
  - Compile list of most probable passwords
  - Apply hash function to each
  - Compare against the password file
  - *If match, password has been found!*
Defending with Salts!

Salt: per user random value

Password file

1. Login A, pwd

<table>
<thead>
<tr>
<th>User ID</th>
<th>salt</th>
<th>H(salt, pwd)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>s</td>
<td>$H_A$</td>
</tr>
</tbody>
</table>

$H(s, pwd) == H_A$
### Example: Linux

- Passwords stored in `/etc/shadow`
  - Root readable only
- carbunar: $6$IGHQQKZn$8.eJLvAaJiDTFAauGVbFlmnAcjKIyLtH6GiO0mVgra8weKJ1igU2BmgdDQAalynFQ0QuezQr7mDTWEPD7sDrW
- **$6**: hash algorithm
  - $1$ = MD5 hashing algorithm.
  - $2$ = Blowfish Algorithm is in use.
  - $2a$ = eksblowfish Algorithm
  - $5$ = SHA-256 Algorithm
  - $6$ = SHA-512 Algorithm
Example: Linux

- Passwords stored in /etc/shadow
  - Root readable only
- carbunar:$6$lGHQQKZn$8.eJLvAaJiDTFAauGVbFlmnAcjIKyLtH6GiO0mVgra8weKJ1igU2BmgdDQAalynFQ0QuezQr7mDTWEPD7sDrW
  - salt
  - hash
The Goal of Salts

- Ensure that attacker cannot use the same dictionary to break all passwords

- Instead, attacker has to do a per-user dictionary + computation ...
Improved Dictionary Attack [D. Klein]

1. Copy the password file

2. For each user $A$ with salt $s$ and hash $H_A$
   1. Collect dictionary $D_A$ of tentative passwords
   2. Hash all items in $D_A$ using salt $s$
   3. Compare result against $H_A$

3. If match exists, found password

- 40% of passwords were guessed on average system!
Building the Dictionary

1. Name, initials, account name
   - Example: Daniel V. Klein, account – klone
   - klone0, klone1, ..., dvk, dklein, DKlein, dvklein, etc

2. Words from databases
   - Men and women names, nicknames (also famous)
   - Places
   - Variations of the above (capitalizations, plurals, etc)

3. Foreign language words

4. Word pairs
Conclusions

- Never use your personal information
- Do not use words (dictionary)
- Use combination of words and characters
- Do not use same passwords for all systems
- Change your password frequently
- Use passphrases
- Example:
  - ”My Password is not easy to crack”
  - mpine2C.
**SKEY: Authentication for Machines**

*Use hash-chains*

**Alice**

Generate $R$

**Compute**

$x_1 = H(R)$

$x_2 = H^2(R) = H(H(R))$

$x_3 = H^3(R) = H(H(H(R)))$

$\cdots$

$x_{100} = H^{100}(R)$

**Trent T (Host)**

1. **Init, A, $x_{100}$**

2. **Login, A, $x_{99}$**

3. **Login, A, $x_{98}$**

**Store $x_{100}$**

**Compare**

$H(x_{99})$ to $x_{100}$

**Discard $x_{100}$ Store $x_{99}$**
Authentication

- One-way authentication
- Mutual: two-way authentication
  - Using symmetric key crypto
  - Using public-key crypto
What is Mutual Authentication?

Alice

1. Authenticate

1’. Exchange keys

Bob B

Mallory

Make sure they don’t talk to Mallory!
Authentication

- One-way authentication
- Mutual: two-way authentication
  - Using symmetric key crypto
  - Using public-key crypto
Using Symmetric Keys

1. Exchange keys
1'. Authenticate

Assume T shares a key with A ($K_A$) and B ($K_B$)

$E_A(M)$: encryption with key shared by A and T
Wide-Mouth Frog

Simplest Authentication/Key Exchange

1. Generate random $K$

2. $A, E_A(T_A, B, K)$

3. Decrypt message using $K_A$

4. $E_B(T, A, K)$

5. $E_K(M)$
Wide-Mouth Frog Observations

- Alice and Bob trust each other because of Trent
- Timestamps prevent replay attacks (Why ?)
- Trent is single point of failure/bottleneck
- Assumption:
  - Alice is able to generate good random numbers