CyberSecurity: Principle and Practice

BSc Degree in Computer Science
2022-2023

Lesson 6: User Authentication

Prof. Mauro Conti
Department of Mathematics
University of Padua
conti@math.unipd.it
http://www.math.unipd.it/~conti/

Teaching Assistants
Pier Paolo Tricomi
pierpaolo.tricomi@phd.unipd.it
Tommaso Bianchi
tommaso.bianchi@studenti.unipd.it

UNIVERSITÀ DEGLI STUDI DI PADOVA

SPRITZ SECURITY & PRIVACY RESEARCH GROUP

DIPARTIMENTO MATEMATICA

Introduction

- Fundamental security building block
  - Basis of **access control & user accountability**
- User auth. is the process of **verifying** an identity claimed by or for a system entity
- Has two steps:
  - **Identification** - specify identifier
  - **Verification** - bind entity (person) and identifier
- Distinct from message authentication

Introduction

- User authentication example:
  - User real name: *Alice Toklas*
  - User ID: *ABTOKLAS*
  - Password: A.df1618hJb
- These informations are stored in a system
  - Only Alice can access with this credential
  - But attackers can still do something ...

Different ways to authenticate

- Four means of authenticating user's identity

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

- Can use alone or combined

- All can provide user authentication

- All have issues

Password Authentication

- Widely used user authentication method
  - User provides name/login and password
  - System compares password with that saved for specified login
- Authenticates ID of user logging and
  - That the user is authorized to access system
  - Determines the user’s privileges
  - Is used in discretionary access control

Password Vulnerabilities

- Offline dictionary attack
  - **Attack:** the attacker has the hash of the target password and he tries to break it
    - Common passwords
    - Info related to the target
  - **Countermeasure**: ?

Password Vulnerabilities

- Offline dictionary attack
  - **Attack:** the attacker has the hash of the target password and he tries to break it
    - Common passwords
    - Info related to the target
  - **Countermeasure**:
    - Protect these informations

Password Vulnerabilities

- Specific account attack
  - **Attack:** the attacker target a specific account and tries to guess the correct password
    - Common passwords
    - Info related to the target
  - **Countermeasure**: ?

Password Vulnerabilities

- Specific account attack
  - **Attack:** the attacker target a specific account and tries to guess the correct password
    - Common passwords
    - Info related to the target
  - **Countermeasure**:
    - account lockout mechanisms (i.e., allow only few authentication attempts)

Password Vulnerabilities

- Popular Password Guessing
  - **Attack**: the attacker tries popular password against a wide range of accounts
    - Users tend to choose simple passwords
    - Likely to detect some passwords
  - **Countermeasure**: ?

Password Vulnerabilities

- Popular Password Guessing
  - **Attack**: the attacker tries popular password against a wide range of accounts
    - Users tend to choose simple passwords
    - Likely to detect some passwords
  - **Countermeasure**:
    - Policies that do not allow the use of simple and common passwords

Password Vulnerabilities

- Workstation hijacking
  - **Attack**: The attacker waits until a logged-in workstation is unattended
  - **Countermeasure**: ?

Password Vulnerabilities

- Workstation hijacking
  - **Attack**: The attacker waits until a logged-in workstation is unattended
  - **Countermeasure**: **Countermeasure**:
    - Automatically logging-out mechanisms
    - Anomaly behaviour detection

Password Vulnerabilities

- Exploiting user mistakes
  - **Attack**: Users tend to write down passwords
    - E.g., post-it near the protected device
    - Devices with pre-configured passwords
  - **Countermeasure**: ?

Password Vulnerabilities

- Exploiting user mistakes
  - **Attack**: Users tend to write down passwords
    - E.g., post-it near the protected device
    - Devices with pre-configured passwords
  - **Countermeasure**:
    - User training
    - Combined authentication mechanism
      - Password + token

Password Vulnerabilities

- Exploiting multiple password uses
  - **Attack**: users tend to use same (or similar) passwords in different systems
    - If an attacker correctly guess a password, he can extend the damage in multiple systems
  - **Countermeasure**: ?

Password Vulnerabilities

- Exploiting multiple password uses
  - **Attack**: users tend to use same (or similar) passwords in different systems
    - If an attacker correctly guess a password, he can extend the damage in multiple systems
  - **Countermeasure**:
  - User training
  - Forbid the password-reuse in multiple systems
    - Feasible only on a specific network that we can control

Password Vulnerabilities

- Electronic monitoring
  - **Attack**: if the password is communicated through a network, an attacker can sniff these packets and steal the password
  - **Countermeasure**: ?

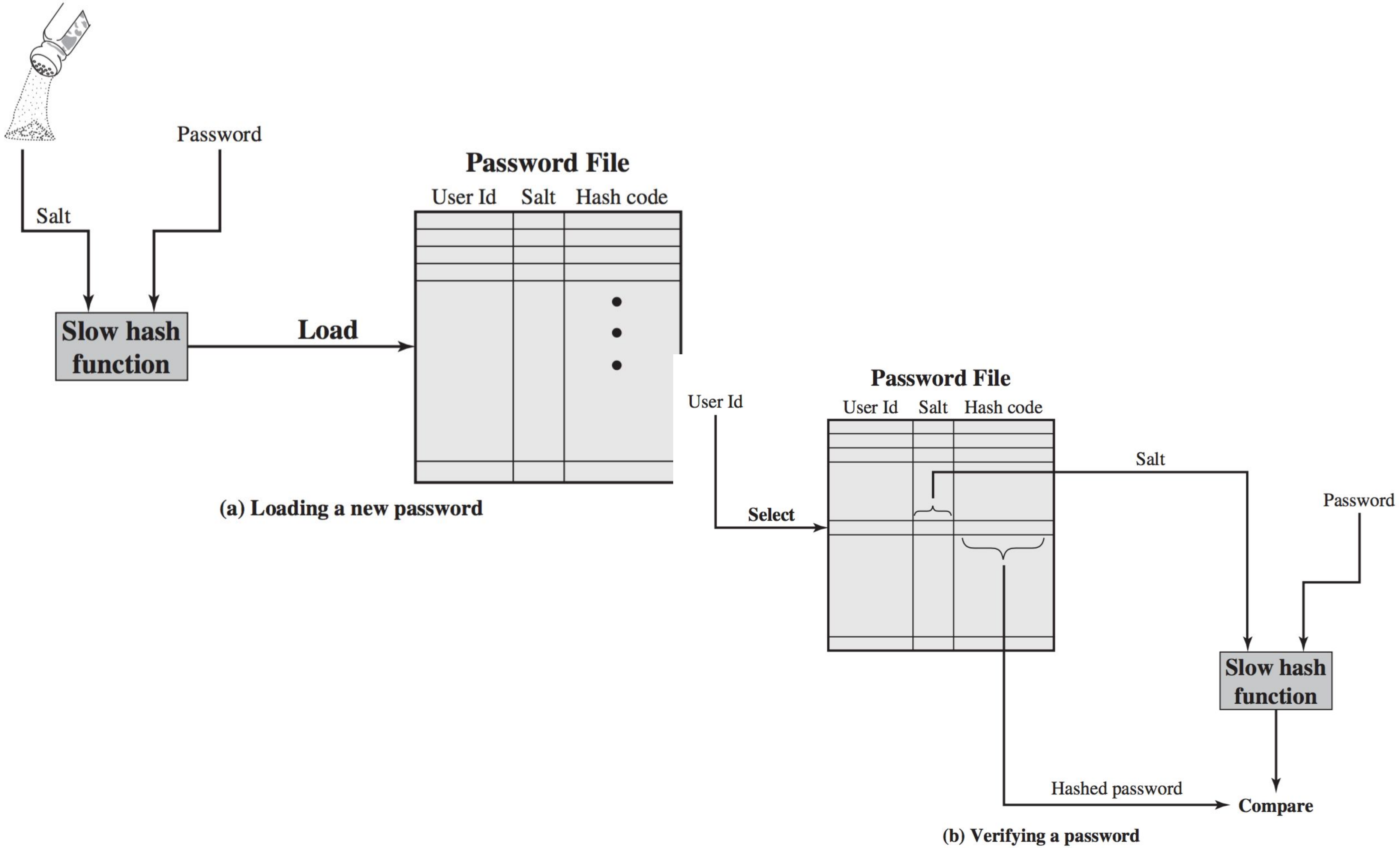
Password Vulnerabilities

- Electronic monitoring
  - **Attack**: if the password is communicated through a network, an attacker can sniff these packets and steal the password
  - **Countermeasure**:
    - Secure communication links

Hashed Passwords

- Widely used security mechanism
- Steps:
    - The user create a new password
    - The password is **combined with a fixed length salt**
        - The salt usually is pseudo-randomly generated
    - Hashed Password = Hash(password, salt)
- ID, Hashed password and Salt are saved in a file
    - Password file
- These hashed functions are designed to be slow

Hashed Passwords



(a) Loading a new password

(b) Verifying a password

Hashed Passwords

- Why do we use salt?
- We can identify three main reasons:
  - Password duplication prevention
    - If two users share the same password, the use of different salt produce different hashed passwords
  - Increase the difficultness of dictionary attacks
    - If the salt has b bits, the factor will be 2^b
  - Impossible to find out if a person uses the same password in different systems

Password Cracking

- Dictionary attacks
  - Try each word then obvious variants in large dictionary against hash in password file
    - First try all common password
    - If there is no-matches, we try possible modifications (numbers, punctuation)
    - Computationally expensive
- Rainbow table attacks
  - Precompute tables of hash values for all salts
  - A mammoth table of hash values
  - E.g., 1.4GB table cracks 99.9% of alphanumeric Windows passwords in 13.8 secs
  - Not feasible if larger salt values used

Password Choices

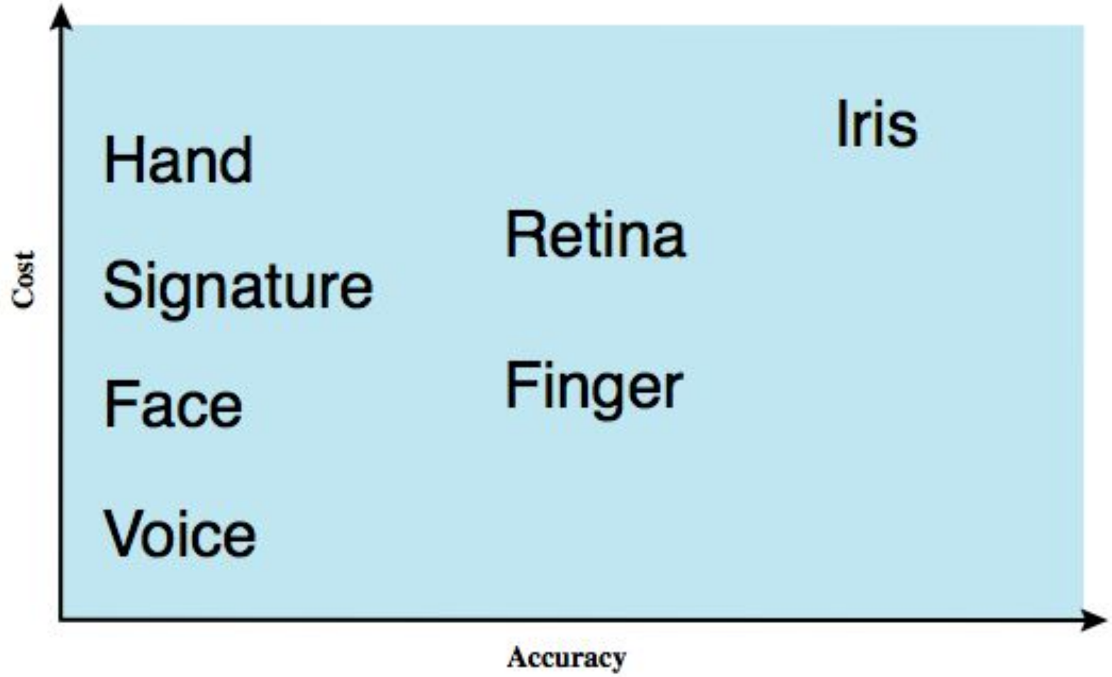
- Users may pick short passwords
  - E.g., 3% were 3 chars or less, easily guessed
  - System can reject choices that are too short
- Users may pick guessable passwords
  - So crackers use lists of likely passwords
  - E.g., one study of 14000 encrypted passwords guessed nearly 1/4 of them
  - Would take about 1 hour on fastest systems to compute all variants, and only need 1 break!

Token Authentication

- Object user possesses to authenticate, e.g.
  - Embossed card (e.g., old credit cards)
  - Magnetic stripe card (e.g., hotel keys)
  - Memory card (e.g., SIM)
  - Smartcard (e.g., Biometric ID card)

Biometric Authentication

- Authenticate user based on one of their physical characteristics



Questions? Feedback? Suggestions?

Memory Card

- BACKUP slides after this point

Memory Card

- Store but do not process data
- Magnetic stripe card, e.g. bank card
- Electronic memory card
- Used alone for physical access
- With password/PIN for computer use
- Drawbacks of memory cards include:
  - Need special reader (increase the cost of the security solution)
  - Loss of token issues (we cannot trust users)
  - User dissatisfaction (not totally approved by users)

Remote User Authentication

- Two type of authentications:
  - Local and from remote
- Authentication over network more complex
  - problems of eavesdropping, replay
- Generally use challenge-response
  - User sends identity
  - Host responds with:
    - Random number *r* (a.k.a. nonce)
    - An hash function *h*
    - A function *f*
  - User computes f(r,h(P)) and sends back
  - Host compares value from user with own computed value, if match user authenticated
- Protects against a number of attacks