User authentication and access control

Mihai Ordean
Designing and Managing Secure Systems
University of Birmingham

Overview

User authentication and access control

- Classification
- Usage
- Password entropy
- Attacks specific to passwords
- Usability issues related to passwords
- Types of passwords

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

1. Something you know
   e.g., user account names and passwords
2. Something you have
   e.g., smart cards or other security tokens
3. Something you are
   e.g., biometrics

Variants of passwords

- Password: a single word,
- Passphrase: a sequence of words or other text used for similar purpose as password
- PIN: a numeric short “password”
- Graphical passwords: an image aided password

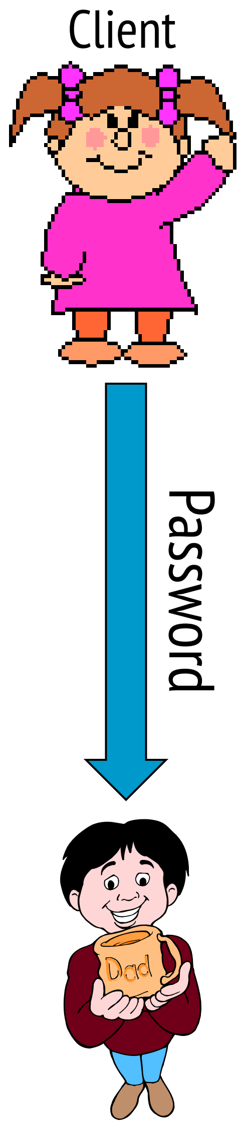
User authentication scenarios

Scenarios

- Logging into a local computer
- Logging into a computer remotely (i.e. from a remote location)
- Logging into a network (i.e. to a remote location)
  - Access web sites

Vulnerabilities can exist at:

- client side
- server side
- communications channel



Password specific attacks

What's your password?!

Password specific attacks

Social engineering

e.g., pretexting: creating and using an invented scenario (the pretext) to persuade a target to release information

Offline dictionary attacks

Online guessing attacks

- exploiting weak passwords
- no rate limiting password input

Password specific attacks

Eavesdropping the communication

e.g. an insecure channel between client and server

Login spoofing

- human errors
- shoulder surfing
- keyloggers

Guessing attacks

1. The average number of guesses the attacker must make to find the correct password (i.e. password entropy)
   - determined by how unpredictable the password is, including how long the password is, what set of symbols it is drawn from, and how it is created.

2. The ease with which an attacker can check the validity of a guessed password
   - determined by how the password is stored, how the checking is done, and any limitation on trying passwords.

Password entropy

- The entropy bits of a password, i.e., the information entropy of a password, measured in bits, is:
  - the base-2 logarithm of the number of guesses needed to find the password with certainty
  - a password with, i.e. 42 bits of strength calculated in this way would be as strong as a string of 42 bits chosen randomly
  - adding one bit of entropy to a password doubles the number of guesses required
  - **on average**, an attacker will have to try half the possible passwords before finding the correct one

Estimating password entropy

People are notoriously bad at achieving sufficient entropy to produce satisfactory passwords.

NIST suggests the following scheme to estimate the entropy of human-generated passwords:

- the entropy of the first character is four bits
- the entropy of the next seven characters are two bits per character
- the ninth through the twentieth character has 1.5 bits of entropy per character
- characters 21 and above have one bit of entropy per character

Estimating password entropy

People are notoriously remiss at achieving sufficient entropy to produce satisfactory passwords.

NIST suggests the following scheme to estimate the entropy of human-generated passwords:

- the entropy of the first character is four bits
- the entropy of the next seven characters are two bits per character
- the ninth through the twentieth character has 1.5 bits of entropy per character
- characters 21 and above have one bit of entropy per character

This would imply that an eight-character human-selected password has about 18 bits of entropy

Better measurements of password entropy?

- NIST suggestion fails to consider usage of different category of characters:
  - lower-case letters, digits, upper-case letters, special symbols
- Orders matters!
  - “Password123!” should have different entropy from “ao3swPd!2s1r”

- Use advanced techniques to compute entropy?
  - An AI classifier (or even a Markov chain model) could rank “kjths” as very secure (approx.. 23 bits of entropy)

- The challenge is to analyze the different attack strategies available to the adversary!

Example of Weak Passwords (from Wikipedia)

- Default passwords (as supplied by the system vendor and meant to be changed at installation time)
  password, default, admin, guest, etc.
- Dictionary words
  chameleon, RedSox, sandbags, bunnyhop!, IntenseCrabtree, etc.
- Words with numbers appended
  password1, deer2000, john1234, etc.
- Words with simple obfuscation
  p@ssw0rd, l33th4x0r, g0ldf1sh, etc.
- Doubled words
  - crabcrab, stopstop, treetree, passpass, etc.

Example of Weak Passwords (from Wikipedia)

- Default passwords (as supplied by the system vendor and meant to be changed at installation time)
  password, default, admin, guest, etc.
- Dictionary words
  chameleon, RedSox, sandbags, bunnyhop!, IntenseCrabtree, etc.
- Words with numbers appended
  password1, deer2000, john1234, etc.
- Words with simple obfuscation
  p@ssw0rd, l33th4x0r, g0ldf1sh, etc.
- Doubled words
  - crabcrab, stopstop, treetree, passpass, etc.

CAN BE TESTED AUTOMATICALLY!

Example of Weak Passwords (from Wikipedia)

- Common sequences from a keyboard row

  qwerty, 12345, asdfgh, fred, etc.
- Numeric sequences based on well known numbers

  911, 314159, or 27182, etc.
- Identifiers

  jsmith123, 1/1/1970, 555–1234, "your username", etc.
- Anything personally related to an individual
  - license plate number, Social Security number, current or past telephone number, student ID, address, birthday, sports team, relative's or pet's names/nicknames/birthdays, etc.

Example of Weak Passwords (from Wikipedia)

- Common sequences from a keyboard row

  qwerty, 12345, asdfgh, fred, etc.
- Numeric sequences based on well known numbers

  911, 314159, or 27182, etc.
- Identifiers

  jsmith123, 1/1/1970, 555–1234, "your username" etc.
- Anything personally related to an individual
  - license plate number, Social Security number, current or past telephone number, student ID, address, birthday, sports team, relative's or pet's names/nicknames/birthdays, etc.

CAN BE TESTED AUTOMATICALLY!
(after a bit of investigating)

Avoiding weak passwords

- Allow long passphrases
- Randomly generate passwords where appropriate
  - though probably inappropriate for most scenarios
- Check the quality of user-selected passwords
  - use a number of "rules of thumb"
  - 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

Password usability

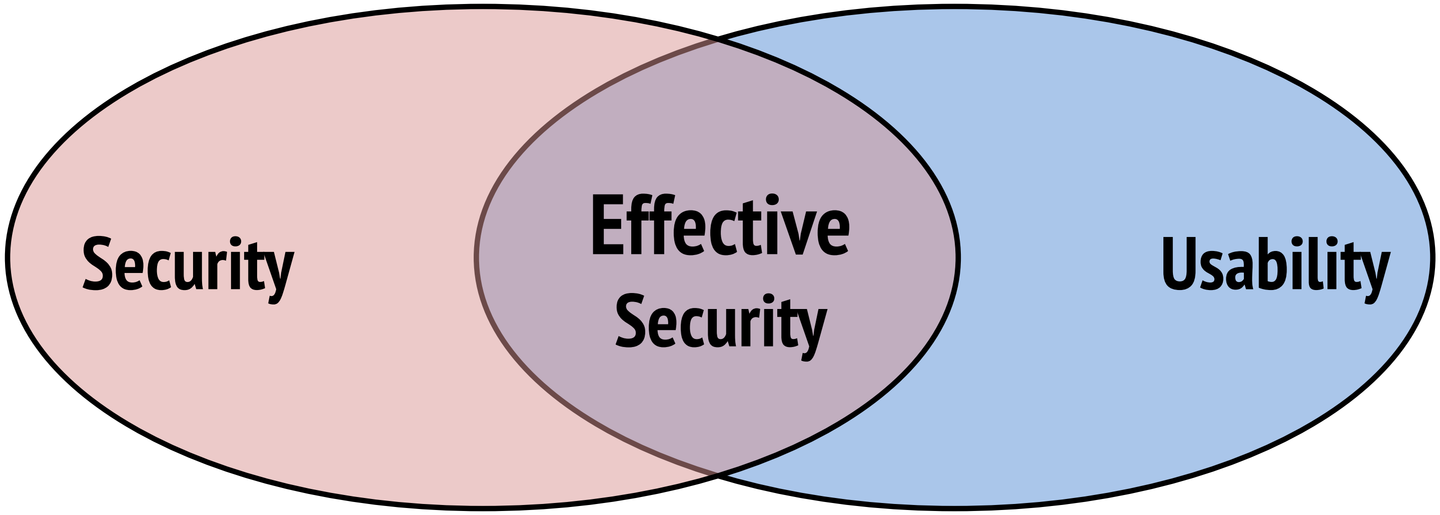
- Forcing randomly generated passwords is often bad.
  - a user needs to remember passwords for tens, if not hundreds of accounts
  - high entropy passwords are difficult to remember
- Guessing passwords is NOT the weakest link
  - various ways exist to reduce adversary's abilities to test password guesses
  - when a user cannot remember the password for an account, there must be a way to allow a user to retrieve it
    - The recovering method either has low security, or costs lots of money
    - It creates a weaker link.

Password usability

- Forcing randomly generated passwords is often bad.
    - a user needs to remember passwords for tens, if not hundreds of accounts
    - high entropy passwords are difficult to remember
- Guessing passwords is NOT the weakest link
    - various ways exist to reduce adversary's abilities to test password guesses
    - when a user cannot remember the password for an account, there must be a way to allow a user to retrieve it
        - The recovering method either has low security, or costs lots of money
        - It creates a weaker link.

Usability matters!

Security vs usability



Password space
Attack resistance
Cryptography

Memorability
Difficulty of use
Efficiency

Password evaluation criteria

Security

Authentication interfaces

- Brute-force attacks
- Dictionary attacks
  - Pattern analysis
  - Statistical analysis
  - Probing
  - Intersection
- Malware
- Shoulder-surfing
- Phishing/Pharming

Authentication protocols

- Replay attack
- Parallel session attack
- Man-in-the-Middle attacks
- Protocol design flaws

Usability

Authentication Interfaces

- Memorability
- User target
- Target domain
- Hardware req.
- Specific procs.:
  - Enrollment
  - Authentication
  - Credential update

Authentication protocols

- Algorithm efficiency
- Computational req.

Mitigating attacks against passwords

Brute-force and dictionary attacks

- Brute-force and dictionary attacks
  - Attacks that leverage low entropy in passwords
- Defences
  - Protect stored passwords (use both cryptography & access control)
  - Disable accounts with multiple failed attempts
  - 2FA: Require extra authentication mechanism (e.g., phone, other email account, etc.)

Login spoofing

- Login spoofing attacks
  - write a program showing a login window on screen and record the passwords
  - put a malicious **su** in the current directory
- Defences
  - Mechanisms that provide confidence that the user is communicating with the real intended server
    - 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
    - Causes a non-maskable interrupt that can only be intercepted by the operating system, guaranteeing that the login window cannot be spoofed

Web based password spoofing

- Phishing attacks
  - attempting to acquire sensitive information such as usernames, passwords and credit card details by masquerading as a trustworthy entity in electronic communication.
- Website forgery
  - Set up fake websites that look like e-commerce sites and trick users into visiting the sites and entering sensitive info
- Defences
  - Browser filtering of known phishing sites
  - Cryptographic authentication of servers (will talk about in future)
  - User-configured authentication of servers
    - Ensures that the site is the one the human user has in mind
    - E.g., site key, pre-selected picture/phrases

Key logging

- Keystroke logging and similar threats from insecure client side
    - Keystroke logging (keylogging) is the action of tracking (or logging) the keys struck on a keyboard, typically in a covert manner so that the person using the keyboard is unaware that their actions are being monitored.
- Software-based
    - key-stroke events, grab web forms, analyse HTTP packets
- Hardware-based
    - Connector, wireless sniffers, acoustic based
- Defences
    - Anti-spyware, network monitors, on-screen soft keyboard, automatic form filler, etc.
    - In general difficult to deal with once on the system

Case studies

UNIX password management

Old UNIX

- The file /etc/passwd stores $H(pwd)$ together with each user's login name, user id, home directory, login shell, etc.
  - H is a one-way hash function
- The file /etc/passwd must be world readable
- Brute force attacks possible even if H is one-way
  - How to prevent **brute-force attacks** on a system with many accounts?

UNIX password management

Modern *NIX

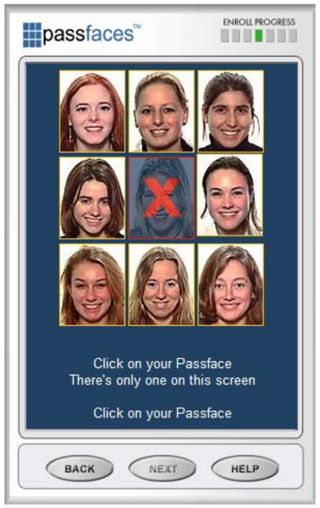
- Divide /etc/password into two files:
  - /etc/password (readable by everyone)
  - /etc/shadow (readable only by root)
- Store $[\mathrm{id}, r, H(r, pwd)]$ rather than $[\mathrm{id}, H(pwd)]$ in /etc/shadow
  - r is **randomly** chosen for each password
  - r is a **public** value, called **nonce**
- Benefits
  - if two users happen to choose the same password, it doesn't immediately show
  - dictionary attacks much more difficult
  - cost of attacking a single account remains the same

Graphical passwords

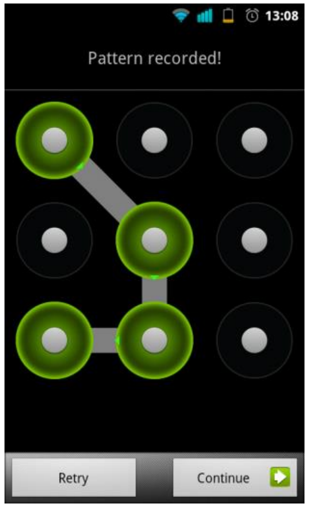
A graphical password is an authentication system that works by having the user select from images, in a specific order, presented in a graphical user interface (GUI). For this reason, the graphical-password approach is sometimes called graphical user authentication (GUA).

Margaret Rouse
https://searchsecurity.techtarget.com/definition/graphical-password

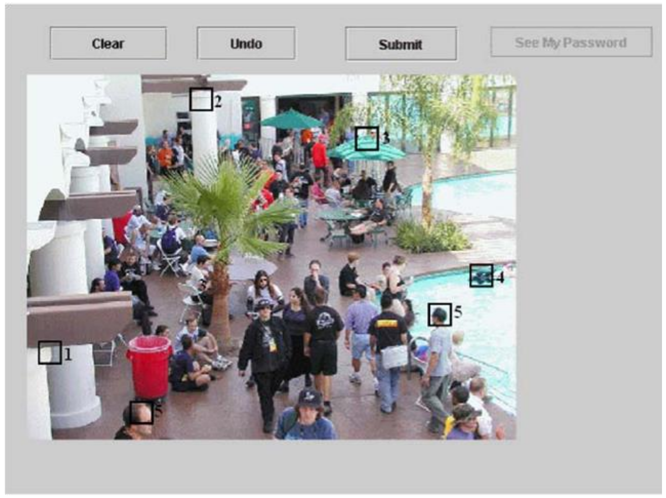
Graphical passwords



Recognition based authentication



Recall based authentication



Cued-recall based authentication

Graphical passwords

- Advantages
  - easier to memorize
  - could have much larger key-space
  - much harder to write down

- Disadvantages
  - much harder to replace
  - additional attacks (e.g. intersection attacks, smudge analysis, hotspots)
  - require GUI i.e. less deployable

Passwords over insecure channels

- One-time passwords
  - each password is used only once
  - defend against passive adversaries who eavesdrop and later attempt to impersonate
- Challenge-response protocols
  - send a response related to both the password and a challenge

One-time passwords

- Shared lists of one-time passwords
- Time-synchronized OTP
  - e.g., use $MAC_K(t)$, where K is shared secret, and t is current time
- Using a hash chain (Leslie Lamport, 1981)
  1. $H(s), H(H(s), H(H(H(s))), \ldots, H^{1000}(s)$
  2. use these values as passwords in reverse order

Hash chain (Lamport's idea)

- Setup:
  1. Alice selects a value $\boldsymbol{p}$ a hash function $\boldsymbol{H(p)}$ and an integer $\boldsymbol{n}$, computes $\boldsymbol{H^n(p)}$ and sends it to Bob
  2. Bob stores $\boldsymbol{H^n(p)}$
- Authentication (round $\boldsymbol{i, 1 \leq i \leq n}$):
  1. Alice sends to Bob: $\boldsymbol{i, p_i = H^{n-i}(p)}$
  2. Bob checks: $\boldsymbol{H(p_i) = H^{n-i+1}(p)}$
  3. If true Bob authenticates Alice and stores: $\boldsymbol{H^{n-i}(p)}$

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 freshness.
  - e.g., nonce (used only once), timestamps

Challenge-response based on symmetric-key crypto

- Unilateral authentication, timestamp-based
  - Alice to Bob: $\boldsymbol{MAC_K(t_A, B)}$

- Unilateral authentication, nonce-based
  - Bob to Alice: $\boldsymbol{n_B}$
  - Alice to Bob: $\boldsymbol{MAC_K(n_B, B)}$

- Mutual authentication, nonce-based
  - Bob to Alice: $\boldsymbol{n_B}$
  - Alice to Bob: $\boldsymbol{n_A, MAC_K(n_A, n_B, B)}$
  - Bob to Alice: $\boldsymbol{MAC_K(n_B, n_A)}$

Wrap up 1: open problems of passwords

- Alternatives to passwords?
  - the secret should be **easy to remember**, **difficult to guess**, and **easy to enter into the system**.
- Better ways to make user choose stronger passwords?
- Better ways to use other devices for authentication
- Effective 2-factored and/or out of band authentication for the Web
- Phishing defense

Wrap up 2: alternatives to “something you know”

Going beyond passwords:

- security tokens
- biometrics
- 2-factor authentication
  - uses two independent authentication methods
  - out of band authentication: uses a channel other than the internet e.g., phone