Josh Wilcox (jw14g24)

March 14, 2025

Table of Contents

Methods of User Authentication

- User authentication is the **primary line of defence** to accessing a computer system
- Authentication is the determination of the identity of any user
- Encompasses two functions:
  - **Identification** - User identifies themselves to a system by presenting credentials
  - **Verification** - System verifies the user by the exchange of auth informations
- Authentication these processes (alone or in combination)
  - Something the individual **Knows**
    - Passwords
  - Somethign the individual **Posesses**
    - USB Keys
    - 2FA Devices
  - Something the individual **Is**
    - Fingerprints, Facial and Iris recognition etc.
    - **Static Biometrics**
  - Something the individual does
    - Voice, Written Signature etc
    - **Dynamic Biometrics**

Password-Based Authentication

- The authentication that is **widely used**
  - User provides username and passwords
  - System compares password to a previously stored one in a password file
- Authenticates the ID of the individual logging into the system

Password-Based Authentication - **Drawbacks**

- Passwords can be predictable
- Users may reuse passwords
- Data breaches may leak passwords

Token Based Authentication

- Tokens are **objects that a user posesses** for auth purposes
- Includes the use of:
  - **Memory Cards** - Stores data but doesn't process data
    - E.g. Old bank cards with a magnetic stripe, can be read and overwrriten by inexpensive card readers
  - **Smart Card** - Has a microprocessor to process data
    - Stronger, can use challenge-response authentication protocol

Token Based Authentication - **Drawbacks**

- You can lose the token
  - Administrative costs in replacing lost token
  - Prevents user from gaining system access
  - If stolen, an adversary could gain system access
- Can be quite inconvenient - Have to remember your token

Biometric Authentication

- Authentication method based on the unique physical characterisitcs of a user
  - **Static** - Fingerprints, facial characteristics, iris patterns
  - **Dynamic** - What the voice sounds like or a written signature
- Based particularly on **pattern recognition**
  - Physical characteristics are mapped into a digital representation
  - Auth system compares stored representation to presented representation
    - Uses a matching score (similarity)

Biometric Authentication - **Drawbacks**

- False Matches
  - False positive
  - Authenticate an **imposter** (sus)
- False NonMatches
  - False Negative
  - Can fail to auth a genuine user
- This concept of accuracy of a biometric system does not apply in passwords and tokens

Remote User Authentication

- Adversaries can eavesdrop authentication processes and hijack the process
- This can be solved by **Challenge-Response** Mechanisms
  - The system sends a challenge (a random value) to the user
  - The user encrypts the challenge with a secret key and sends it back
  - The system decrypts the response and verifies it matches the original challenge
  - Prevents replay attacks as the challenge is different each time

Multi-Factor Authentication

- User is granted a system after successfully presenting **two or more pieces of evidence**
- Typically used with OTP systems where the user can find a code
- Makes it very hard for the adversary

Password Cracking

Brute Force Attacks

- Could use an **Exhaustive Search**
  - Try all possible combinations of symbols up to a certain length
  - For the alphabet set A and a password of length n - the number of passwords to try is

$$|A|^n$$

- Password of length 8 has 96^8 password combinations (7.2 quadrillion)

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years

Dictionary Attacks

- Attempt passwords associated with the user
  - From data leaks
  - Names, Pets, Name of friends, etc.
- Try words in a dictionary
- Try common passwords (12345678)

Password Cracking Countermeasures

- **Choosing Passwords**
  - Long Password Length
  - Mix upper and lowerr case, numbers and symbols
  - Avoid obvious passwords
- Use a password manager
- Change passwords regularly
- Lock account after unsuccessful attempts
- Enforce time delays between failed attempts

Offline Dictionary Attacks

- Determined hackers can find access to the password file **on the system**
- Can be done by comparing the password hashes against hashes of commonly used passwords

Rainbow Tables

- A rainbow table is a precomputed table for reversing cryptographic hash functions
- Used to crack password hashes
- Contains a large set of possible plaintext passwords and their corresponding hash values
- Allows attackers to quickly look up the plaintext password for a given hash
- Reduces the time needed to crack a password by trading off memory for computation
- Can be mitigated by using salts, which add random data to passwords before hashing

Rainbow Tables - **How it works**

1. **Hashing Process:** Given a plaintext password P, a cryptographic hash function H is applied:

$$H(P) = h_1$$

where h_1 is the resulting hashed value.

2. **Reduction Function:** Instead of storing all hashes directly, a reduction function R maps the hash back to another potential password:

$$R(h_1) = P_2$$

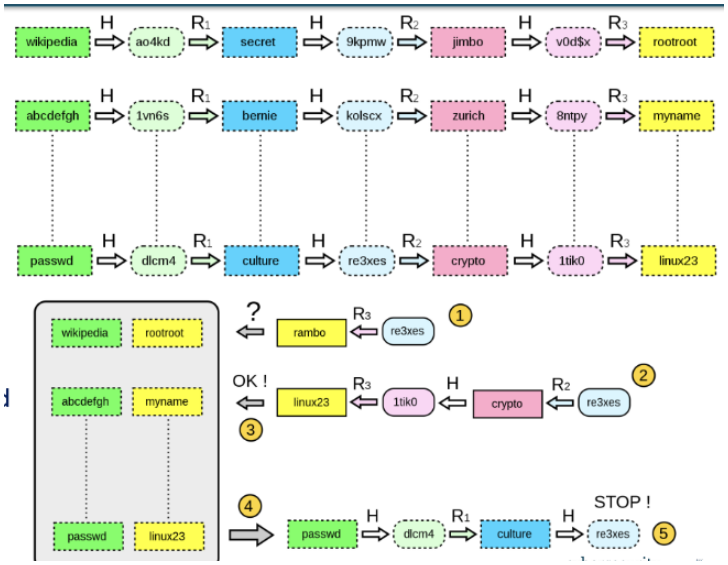
This allows chaining, where the password is hashed again:

$$H(P_2) = h_2, \quad R(h_2) = P_3, \quad \text{etc.}$$

3. **Building Chains:** This process repeats multiple times, forming chains of transformations from an initial password P to a final hash h_n. Only the first and last values of the chain are stored to save space.

4. **Cracking a Password:** If an attacker obtains a hashed password h_{target}, they search for it in the final column of the rainbow table. If found, they trace the chain back to determine the original password.

Rainbow Tables - **Example Attack**



- The chains are built from initial plaintext passwords (green) and transformed through multiple hashing and reduction steps.
- The attacker, given a hashed value (e.g., "re3xes"), looks for it in the table.
- If found, they use the precomputed chain to determine the original password (e.g., "passwd" $\rightarrow$ "dlcm4" $\rightarrow$ "culture" $\rightarrow$ "re3xes").
- If not directly found, they may need to recompute parts of the chain.

Password Salting

- **Salting** is the process of adding a unique, random value to each password before hashing.
- This ensures that even if two users have the same password, their hashes will be different.
- **How it works:**
  - A salt value is generated for each password.
  - The salt is concatenated with the password.
  - The combined value is then hashed.
  - Both the salt and the hash are stored in the password database.
- **Benefits:**
  - Prevents attackers from using precomputed tables (rainbow tables) to crack passwords.
  - Increases the time and computational resources required for brute-force attacks.
  - Ensures that identical passwords have unique hashes.
- **Example:**
  - Password: `password123`
  - Salt: `randomSalt`
  - Combined: `password123randomSalt`
  - Hash: `H(password123randomSalt)`