Authentication Technologies

- HTML forms-based authentication
- Multifactor mechanisms, such as those combining passwords and physical tokens
- Client SSL certificates and/or smartcards
- HTTP basic and digest authentication
- Windows-integrated authentication using NTLM or Kerberos
- Authentication services

- **Over 90% of apps use name & password**

More Secure Methods

- **Two-factor authentication (or more)**
  - **PIN from a token, SMS message, or mobile app**
  - **In addition to a password**
  - **Submitted through an HTML form**

Cryptographic Methods

- **Client-side SSL certificate**
- **Smartcards**
- **More expensive, higher overhead**

HTTP Authentication

- **Basic, Digest, and Windows-integrated**
- **Rarely used on the Internet**
- **More common in intranets, especially Windows domains**
  - **So authenticated employees can easily access secured resources**

Design Flaws

- **Bad Passwords**
  - Very short or blank
  - Common dictionary words or names
  - The same as the username
  - Still set to a default value

Hack Steps

Attempt to discover any rules regarding password quality:

1. Review the website for any description of the rules.

2. If self-registration is possible, attempt to register several accounts with different kinds of weak passwords to discover what rules are in place.

3. If you control a single account and password change is possible, attempt to change your password to various weak values.

Brute-Force Attacks

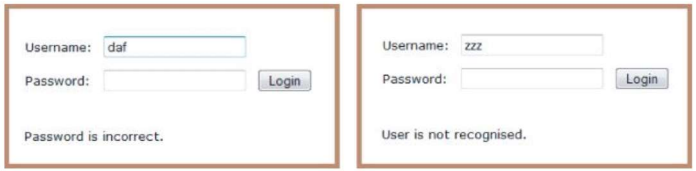
- **Strictly, "brute force" refers to trying every possible combination of characters**
- **Very slow**
- **In practice, attackers use lists of common passwords**
- **Defense: account lockout rules after too many failed login attempts**

- password
- *website name*
- 12345678
- qwerty
- abc123
- 111111
- monkey
- 12345
- letmein

Poor Attempt Counters

- **Cookie containing failedlogins=1**
- **Failed login counter held within the current session**
  - **Attacker can just withhold the session cookie to defeat this**
- **Sometimes a page continues to provide information about a password's correctness even after account lockout**

Verbose Failure Messages



- **Friendly for legitimate users**
- **But helpful for attackers**

Username Importance

- **Attacks that reveal valid usernames are called "username enumeration"**
- **Not as bad as finding a password, but still a privacy intrusion**
- **But could be used for social engineering attacks, such as spearphishing emails**

"Remember Me"

- **Sometimes a simple persistent cookie, like**
  - `RememberUser=jsmith`
  - `Session=728`
- **No need to actually log in, if username or session ID can be guessed or found**

Securing Authentication

- **Considerations**
  - **How critical is security?**
  - **Will users tolerate inconvenient controls?**
  - **Cost of supporting a user-unfriendly system**
  - **Cost of alternatives, compare to revenue generated or value of assets**

Strong Credentials

- **Minimum password length, requiring alphabetical, numeric, and typographic characters**
- **Avoiding dictionary words, password same as username, re-use of old passwords**
- **Usernames should be unique**
- **Automatically generated usernames or passwords should be long and random, so they cannot be guessed or predicted**
- **Allow users to set strong passwords**

Handle Credentials Secretively

- **Protect them when created, stored, and transmitted**
- **Use well-established cryptography like SSL, not custom methods**
- **Whole login page should be HTTPS, not just the login button**
- **Use POST rather than GET**
- **Don't put credentials in URL parameters or cookies**

Hashing

- **Password hashes must be salted and stretched**
- **Salt: add random bytes to the password before hashing it**
- **Stretched: many rounds of hashing (Kali Linux 2 uses 5000 rounds of SHA-512)**

Handle Credentials Secretively

- **Client-side "remember me" functionality should remember only nonsecret items such as usernames**
- **If you allow users to store passwords locally, they should be reversibly encrypted with a key known only to the server**
  - **And make sure there are no XSS vulnerabilities**

Handle Credentials Secretively

- **Force users to change passwords periodically**
- **Credentials for new users should be sent as securely as possible and time-limited; force password change on first login**
- **Capture some login information with drop-down lists instead of text fields, to defeat keyloggers**

Validate Credentials Properly

- **Validate entire credential, with case sensitivity**
- **Terminate the session on any exception**
- **Review authentication logic and code**
- **Strictly control user impersonation**

Multistage Login

- **All data about progress and the results of previous validation tasks should be held in the server-side session object and never available to the client**
- **No item of information should be submitted more than once by the user**
- **No means for the user to modify data after submission**

Prevent Brute-Force Attacks

- **Consider this type of attack**
  - **Use many different usernames with the same password, such as "password"**
- **Defenses: strong password rules, CAPTCHA**

Password Change

- **No way to change username**
- **Require user to enter the old password**
- **Require new password twice (to prevent mistakes)**
- **Same error message for all failures**
- **Users should be notified out-of-band (such as via email) that the password has been changed**

Account Recovery

- **For security-critical apps, require account recovery out-of-band**
- **Don't use "password hints"**
- **Email a unique, time-limited, unguessable, single-use recovery URL**

Account Recovery

- **Challenge questions**
  - **Don't let users write their own questions**
  - **Don't use questions with low-entropy answers, such as "your favorite color"**

Log, Monitor, and Notify

- **Log all authentication-related events**
  - **Protect logs from unauthorized access**
- **Anomalies such as brute-force attacks should trigger IDS alerts**
- **Notify users out-of-band of any critical security events, such as password changes**
- **Notify users in-band of frequent security events, such as time and source IP of the last login**