Attacking Authentication

- On the face of it Authentication is a very simple concept involving just a username and password.
- But this serves as the first line defense of the web app, in real world application authentication is often the weakest link.
- Hence breaking this part of the application can lead to a myriad of issues in the web app

- Authentication Technologies
  → HTML Form base
  → Multifator mechanisms
  → Client SSL certs / smartcards
  → HTTP basic and digest auth
  → Windows integrated authentication using NTLM adn Kerberos
  → Authentication services

- HTML form based authentication accounts for about 90% web apps.

- **Design Flaws in Authentication Mechanisms**

  → Bad Passwords
    ⇒ Employing weak password policies such as,
    • Very short or blank password
    • Dictionary words
    • Same as username
    • Default credentials

  → Brute-Forcible Login
    ⇒ Allowing the server to process to interact with multiple login requests can enable an attacker to guess the password of users using a brute force attack
    ⇒ To mitigate, introduce a cookie that can keep track of failedlogins or attempt to login. If a threshold is reached, the server can refuse to process requests and block the attacker. But ofcourse, this is not absolute defense, because this can be bypassed since it is only employed on client side.
    ⇒ Enforce a tight account lockout policy to mitigate brute force attempts.
    ⇒ Anti-Bruteforcing measure can be bypassed by obtaining new sessions, or sometimes even be enumerated using weird results from the app.

  → Verbose Failure Messages
    ⇒ Error messages that are too particular about the submitted values can lead the attacker to enumerate infomation from the application
    ⇒ Ex: Username Enumeration in a sigup portal
    ⇒ Even if the application's request are identical for valiid and invalid users, the time taken for responding to an attempt can be used to extraploate results.

  → Vulnerable Transmission of Credentials
    ⇒ If an application sends senesitive data over unencrypted HTTP, the data is open to any eavesdropper positioned in the network.
    ⇒ They can be in places like,
    • On the user's local network
    • Within the user's IT dept.
    • Within user's ISP
    • On the internet backbone
    • ISP hosting the web app
    ⇒ Even if HTTPS is implemented if the data is handled in an unsafe manner it can be disclosed
    • If data is sent in parameters of GET req instead of body of POST request
    • if data is used in parameter to be redirected to some other URL
    • if credentials are stored in persistent cookies that can be used in a replay attack leading to account takeovers

  → Password Change functionality
    ⇒ Although this is a necessary part of an effective authentication mechanism, vulns that are deliberately avoided in the main login/signup function can resurface here.
    ⇒ Since the decision tree of a password change can be quite comprehensive, this can lead to subtle flaws in this mechanism.

  → Forgotten Password Functionality

⇒ Bruteforcing a security question

⇒ Password Hints

⇒ Dsiclosing existing password after a succesful challenge

⇒ Dropping the user into an unauthenticated session after the validation of a security question

⇒ Sending a verification email to the email chosen by the user rather than the registered e-mail. (Check Cookies, these might transmit the email,
that the recovery email will be sent to, change this to divert the email to a email of your choosing)

→ Remember me functionality

⇒ Although employed for convenience, this can expose a user to an attack

⇒ Simple persisitent cookies can replayed to bypass authentication

⇒ If the persistent cookie is numerical or iterable then an attacker can use combinations to compromise multiple user using an intercepting proxy
without ever knowing their credentials

⇒ Even if the persistent cookie is cryptic, it can be harvested and replayed using an XSS.
hence vulns can be chained to increase the impact on the whole

→ User Impersonation functions

⇒ Many application allow admin to impoersonate the users to be able to perform actions using the user's context.

⇒ This functiionality can be hijacked
- "Hidden" function that can be guessed or even seen in javascript files
- Modification of cookies that allow impersonation
- Logic errors that when admin impersonation is allowed can lead to a vertical privilege escalation
- Master Password or Backdoor Passwords that can be replayed from a previous brute force attack.

→ Incomplete validation of credentials

⇒ Validating parts of the password

⇒ Case-insensitive validation

⇒ Stripping unusual characters (sometimes on the pretext of input sanitization).

→ Nonunique Username (Rare ***)

⇒ If application allows users to register using same username,
- It can eventually lead to an account takeover as it might reveal the credentials of the account sharing the same username
- Result in access of the other accounts that share the username
- Badly designed self-registration can also allow user-name enumeration

→ Predictable Usernames

⇒ Username that are predictable i.e iterable or sequntials can lead to username numeration with minimal interaction

→ Predicatble Initial Passwords (Common in Intranet Setup)

⇒ Accounts are that are protected by a password that is same for a group of people, or is someway related to their work or personal details
can be guessed.

⇒ Also Initial password can be bruteforced if they are simple and unchanged by the user after issue.

→ Insecure Distribution of Credentials

⇒ Sending the credentials of users using an insecure conduit can result in exposure of this data.

⇒ Account Activation URL that manifest a sequence can be guessed to hijack an account.

⇒ Ex. Webapp that sends users credentials to the user's email "for future reference".

- Implementation Flaws in Authentication

→ Even well designed mechanism can be toppled by a flawed implementation.

→ Fail-Open Login Mechanisms

⇒ A series of logic flaws can result in serious consequences

⇒ An authentication that fails can reveal some or all data of users.

→ Defects in Multistage Login Mechanisms

⇒ Some application employ authentication in mutliple stages

⇒ These stages involves various security checks to validate a user.

⇒ But the logic of these stages can be flawed that can lead to various vulns

⇒ The mulistage approach can lead to a less secure application than a single authentication mechanism

• An appllication can assume that a user that accesses a later stage has already cleared previous stages , hence bypassing this later stage allows the attacker to gain access easily.

• Trusting data that is passed down through stages, often data that is being passed down into stages can be modified in between to allow the attacker to gain access using partial credentilas or use it to escalate the priveleges

• Assuming that the same user identity is being used to complete each stage

◇ An attacker can use valid credentials to cllear early stage and use the later stages to gain access to a victim's account.

→ Insecure Storage of Credentials

⇒ Cleartext or simple cryptic methods such as Base64, MD5 and SHA 1 can lead to exposure of credentials

- Securing Authentication

→ Using String Credentials

⇒ String password policies are to be enforced

⇒ Unique Usernames

⇒ System generated strings are o be generated with sufficient entropy

⇒ Users should be allowed and encouraged to set long and eclectic passwords.

→ Handle Credentials Sev erely

⇒ All credentials should be stored and transmitted in a cryptic manner that also doesnt allow unauthorized disclosure

⇒ Client-Server comm. to be encrypted using tech like SSL

⇒ Only POST req is to be used for sensitive data; Never send sensitive data in parameters in the URL or as cookies(even ephemeral ones).

⇒ All server side application components should store credentials in an irrecoverable manner

⇒ "Remember Me" functions should only store non sensitive data

⇒ A password change facility should be implmented and users are to be enjoined to change their passwords

→ Validate credentials properlly

⇒ Passwords shoudle be validated in full - Case-sensitive, without filtering, modification and truncation

⇒ Aggressive defense against unexpected events

⇒ Pseudo-Code and Source Code to identify logical errors

⇒ Strict user impersonation rules

⇒ String multistage authentication

→ Prev ent Information Leakage

⇒ The various authentication mechanisms shouldnt reveal information about erros

⇒ A single code component for all error handling related to authentocation

⇒ Account-lockout mechanism shouldnt be verbose about its policy

⇒ For self registration,

• The app can create unique random username

• The app can use email address to which further registration measure are mailed to

→ Prev ent Brute-Force Attacks

⇒ Measures for preventing automated attacks should be enforced in all features involving authentication

⇒ Using unpredictable username to avoid enumeration

⇒ Account-lockout should be inn timeframe rather than outright disbandment.

⇒ If temp disablement is implied,

• Prevent information leakage that leads to username enumeration

• The policy's metrics shouldnt be revealed. Use a equivocated message like "Try again later.".

• If account is suspended then attempts to login should be rejected without credential check

• Use CAPTCHA to discourage automated attacks (remember to make captcha a required param while processing requests)

→ Prev ent Misuse of the Passw ord C hange F unction

⇒ A password change functionality should be implemented, users should be encouraged to change their credentials often.

⇒ The function should be accessible only from within an authenticated should.

⇒ No facility should be present to provide a username in the req as a form field or cookie. Allowing this can lead to attackers changing users' passwords.

⇒ Users should be notified out-of-band (email,sms) that their credentials have been altered.

→ Prevent Misuse of the account Recovery Function

⇒ Avoid Password hints as these can allow the attacker to trawl for easy password victims.

⇒ Best measure is to email the user a time limited recovery email, after succesful change another email should notify the user that their password has been changed.

⇒ If secondary challenge is implemented,

- Question Pool should be same for everyone, as user might implement a challenge whose answer is easy to guess for the attacker.
- Responses to challenges should be sufficientl entropic
- Account suspension to prevent brute force
- App shouldnt leak information in the even of failed responsed to challenge (like validity of username)
- After the challenge the user should be sent an email for recovery.
- Avoid directly dropping into authenticated session or alowing them to change the password immediately

→ Log monitor and notify

⇒ The application should log all authentication related events, includding login, logout suspension etc.

⇒ Both succesful and failed attempts to be logged

⇒ Anomalied to be processed and notified to app admins

⇒ Users should be notified out-of-band of any critical security events

⇒ Users should be notified of frequentl occuring security events. For exmaple last login source/IP , geolocation, failed login attempts etc.