# Identification and Authentication Failures

**Identification and Authentication Failures** occur when applications **incorrectly implement or manage user authentication or session controls**, allowing attackers to:

* Compromise passwords
* Take over user accounts
* Bypass login entirely
* Abuse session management (session fixation, missing timeouts)

These vulnerabilities affect **login forms, password reset mechanisms, tokens, cookies, MFA**, and even **API keys**.

An application allows weak passwords like 123456 or doesn't enforce session expiration after logout.

# Authentication vs Identification

| Concept | Description |
| --- | --- |
| Identification | Claiming an identity (e.g., entering a username or email) |
| Authentication | Proving that identity (e.g., password, OTP, biometric) |

# Types of Identification and Authentication Failures

## 1. Weak or Missing Password Policies

* **Example**: 123456 or admin1 allowed
* **Mitigation**: Enforce complexity, minimum length, blacklists (e.g., HaveIBeenPwned)

## 2. Brute Force or Credential Stuffing

* **Example**: No rate limiting or account lockout on login
* **Mitigation**: CAPTCHA, lockout policies, rate-limiting, MFA

## 3. Session Fixation

* **Example**: Same session ID reused before and after login
* **Mitigation**: Regenerate session ID upon authentication

## 4. Exposed Session IDs

* **Example**: Session ID in URL: /home.jsp?JSESSIONID=123456
* **Mitigation**: Use secure, HTTP-only cookies only

## 5. Missing Multi-Factor Authentication (MFA)

* **Example**: High-privilege accounts rely only on passwords
* **Mitigation**: Enforce MFA on admin, finance, sensitive accounts

## 6. Improper Password Storage

* **Example**: Plaintext or unsalted SHA-1 hashes
* **Mitigation**: Use bcrypt, scrypt, or Argon2 with proper salt

## 7. Forgot Password Vulnerabilities

* **Example**: Predictable or guessable password reset tokens
* **Mitigation**: Use long, cryptographically secure, one-time reset tokens

## 8. Auth Logic Bypass

* **Example**: Accessing /admin directly without login
* **Mitigation**: Implement proper access control checks server-side

# Impact

| Area | Risk |
| --- | --- |
| Account Takeover | Identity theft, impersonation |
| Unauthorized Access | Privilege escalation, access to restricted data |
| Business Impact | Trust loss, financial loss, regulatory fines |
| Authentication Bypass | Skip login process completely |
| Session Hijacking | Attacker reuses or fixes session tokens |

# Testing Techniques

| Method | Description |
| --- | --- |
| Brute Force / Dictionary Attack | Try common credentials with tools like Hydra or Burp |
| Password Policy Testing | Try short, simple, or known breached passwords |
| Session Testing | Check if session IDs are predictable, reused, exposed |
| Forgot Password Logic | Inspect reset link/token behavior and expiry |
| MFA Bypass | Try OTP reuse, replay, backup bypass |
| Role Tampering | Check if roles can be modified in JWTs or cookies |

# Tools

| Tool | Purpose |
| --- | --- |
| Burp Suite | Manual testing of login, reset, MFA, sessions |
| Hydra / Medusa | Credential brute-forcing |
| Wfuzz / ffuf | Username/email enumeration |
| jwt.io | Decode JWT tokens, verify signature settings |
| OWASP ZAP | Passive scanning of login/session issues |
| Sn1per / Nuclei | Identify exposed login portals with weak configurations |

# Mitigation:

## Password Security

* Enforce minimum complexity: length ≥ 12, uppercase/lowercase, symbols
* Ban breached passwords (use APIs like HaveIBeenPwned)
* Hash with bcrypt, Argon2, or scrypt

## Session Management

* Use secure, HTTP-only, SameSite cookies
* Regenerate session ID on login/logout
* Enforce session timeout (15–30 mins of inactivity)
* Invalidate tokens on logout or password change

## Account Security

* Implement **rate limiting** and **CAPTCHA** on login
* Enforce **account lockout** on multiple failures (temporary)
* Use **MFA** on all sensitive and high-privilege accounts

## Token Security

* Use cryptographically secure, unique tokens for password reset
* Set short token expiration times (≤ 10 mins)
* Use secure and SameSite flags on all auth cookies
* Enforce strong password policies (length, complexity).
* Implement multi-factor authentication (MFA).

Ensure proper session management, including timeout and invalidation after logout

# Points

“Auth failures are **gateway vulnerabilities** — they don’t break into the system, they log in.”

“Always check **session fixation, password policy enforcement, MFA, and reset token design**.”

“Mitigating auth failures isn’t about code complexity — it’s about strict security hygiene.”

# Real-World Breaches

| Case | Cause |
| --- | --- |
| Facebook (2019) | Stored plaintext passwords internally |
| GitHub (2012) | Password brute force with no lockout |
| Uber (2016) | Password reset token abuse |
| Dropbox (2012) | Used password-only authentication — no MFA |