Authentication Flaws

❌ Typical Flaws in Authentication

- Permits brute force or other automated attacks
- Permits default, weak, or well-known passwords
- Uses weak or ineffective credential recovery and forgot-password processes (e.g. "knowledge-based answers")
- Uses plain text, encrypted, or weakly hashed passwords
- Has missing or ineffective multi-factor authentication
- Exposes Session IDs in the URL
- Does not rotate Session IDs after successful login
- Does not properly invalidate Session IDs

Risk Rating

Broken Authentication

Exploitability	Prevalence	Detecability	Impact	Risk
🔴 Easy	🔶 Common	🔶 Average	🔴 Severe	A2
(3	+ 2	+ 2) / 3	* 3	= 7.0

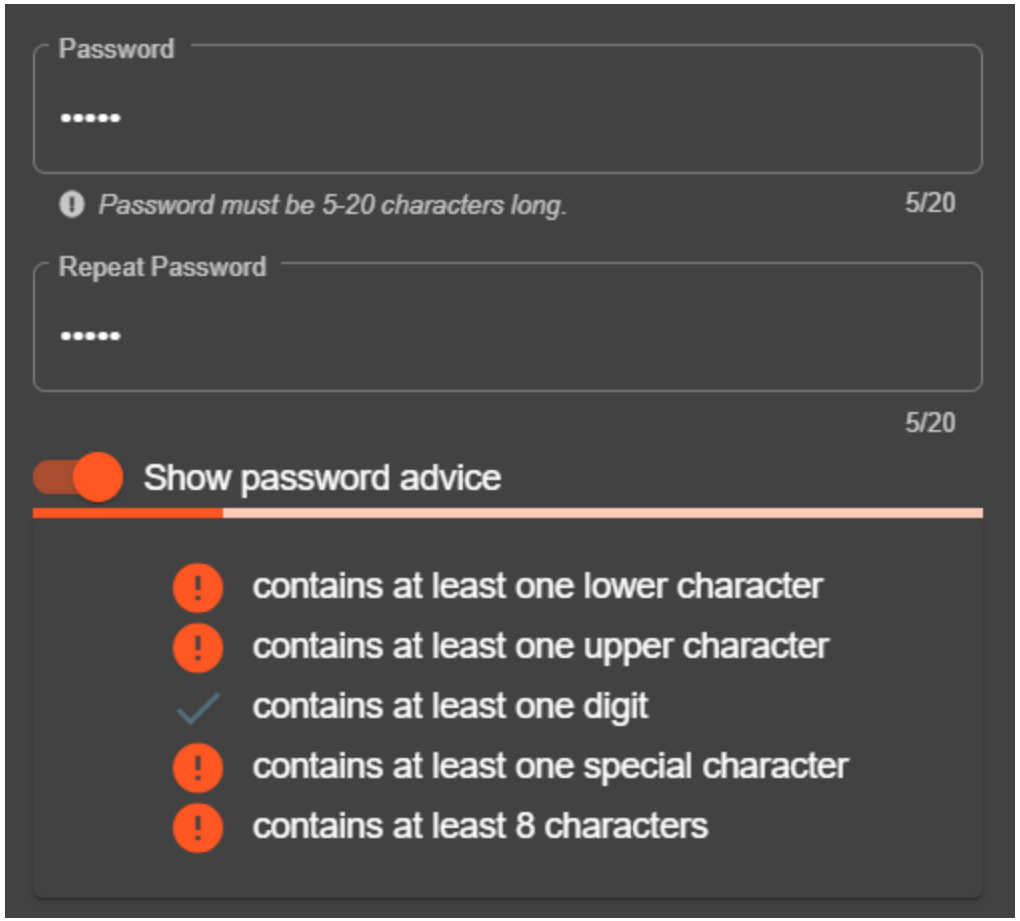
Exercise 4.1 (📺)

1. Watch How To Keep Your Passwords Safe
2. Cast a vote (📩) on which password MC is probably using
3. Log in with MC SafeSearch's user account (⭐⭐)

⚠️ *Do **not** use SQL Injection for authentication bypass!*

Exercise 4.2 (📂)

1. What password might the user currently have typed in?



Prevention

Password Strength Controls

- **Enforce minimum password length** of at least 10 characters
- Maximum length should allow 64 characters or more
- **No periodic password resets** as users rely on predictable patterns
- Avoid password complexity rules as *all of them* are predictable
- Ban bad passwords or ones which have appeared in data breaches
  - e.g. Troy Hunt's 10GB+ list or Daniel Miesler's various lists
- Allow convenience features on password fields
  - Offer *Show Password while typing* option
  - Allow pasting from clipboard into password fields

Secure "Forgot Password" Mechanism

- Return a consistent message for both existent and non-existent accounts
- Ensure that the time taken for the user response message is uniform
- Use a side-channel to communicate the method to reset their password
- Use URL tokens for the simplest and fastest implementation
- Ensure that generated tokens or codes are:
  - Randomly generated using a cryptographically safe algorithm
  - Sufficiently long to protect against brute-force attacks
  - Stored securely
  - Single use and expire after an appropriate period

Secure Password Storage

- Use Bcrypt unless you have a good reason not to
- Set a reasonable work factor for you system
- Use a salt (modern algorithms do this for you automatically)
- Consider using a pepper to provide an additional layer of security

Other Authentication Controls

- **Transmit passwords only over TLS**
  - The "login landing page" must be served over TLS as well
- **Prevent Brute-Force Attacks** (e.g. throttling or periodic lockout)
- Require re-authentication for sensitive features
- **Offer optional 2FA / MFA**
  - Consider strong transaction authentication

Enterprise Controls

- Use centralized corporate authentication system (if in place)

Two-Factor Authentication

> Two-factor authentication adds a second level of authentication to an account log-in. When you have to enter only your username and one password, that's considered a single-factor authentication. 2FA requires the user to have two out of three types of credentials before being able to access an account. The three types are:
>
> - **Something you know**, such as a personal identification number (PIN), password or a pattern
> - **Something you have**, such as an ATM card, phone, or fob
> - **Something you are**, such as a biometric like a fingerprint or voice print [^1]

2FA Method Comparison

Method	Security	Privacy	Access	Prevalence
SMS	🔑	😎	🚪🚪🚪	🏧🏧🏧
Authenticator App	🔑🔑	😎😎😎	🚪	🏧🏧
Hardware Key	🔑🔑🔑	😎😎😎	🚪🚪	🏧

> Hardware keys win from a security perspective, they are private and unaffected by a dying or out of range phone. However, only a few services (Google, Dropbox, Facebook, Github and a few others) support the standard so far. Unless you trust your phone provider (and few providers are trustworthy), **an authenticator app is the best option**.

Password Managers

> Password managers are programs, browser plugins or web services that automate management of large number of different credentials. Most password managers have functionality to allow users to easily use them on websites, either by pasting the passwords into the login form, or by simulating the user typing them in. [^2]

		
Open Source (GPLv2)	Proprietary / Freemium	Proprietary
Local installation, optional file or cloud sync	Cloud-based	Local installation with Cloud sync

> Web applications should at least not make password managers job more difficult than necessary by observing the following recommendations:
>
> - use standard HTML forms for username and password input with appropriate `type` attributes,
> - do not artificially limit user passwords to a length "reasonable for humans" and allow passwords lengths up to 128 characters,
> - do not artificially prevent copy and paste on username and password fields,
> - avoid plugin-based login pages (Flash, Silverlight etc) [^1]

Exercise 4.4 (🏠)

1. Log in with the admin's user account (★★)
2. Reset Jim's password by answering his secret question (★★★)
3. Log in with Bjoern's Google account (★★★★)

⚠️ *Do **not** use SQL Injection for authentication bypass! Also, do **not** hack Bjoern's actual Google account!*

Exercise 4.5 (🏠)

1. Read https://webauthn.guide/ and play with https://webauthn.io/ to learn how WebAuthn works

Exercise 4.6 (*optional*)

1. Install a 2FA app on your phone (e.g. Google Authenticator or Authy)
2. Visit https://twofactorauth.org and find out what services you use offer 2FA
3. Turn 2FA on wherever possible
4. Do not forget to print (= 🖨️!) the backup codes and keep them safe

💡 *Pro tip: Print hard copies of the originally displayed QR codes for easy setup on any new phone! Just store them very securely!*