

# 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. Log in with MC SafeSearch's user account ( )

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

## Exercise 4.2 ()

1. Identify all flaws in the generator of the following session IDs

#	Session ID	#	Session ID
1	h5kek4z9ha1rtrf	7	po953ld7hg2awi9
2	gj75l3k7hb15rtr	8	t6zhj2n5hh27bn0
3	l8l65k45hc1rw7i	9	iu345r53hi2aw34
4	p05jrj53hd1i039	10	o0z43411hj2njkl
5	5urltda1he1bn46	11	9por42o9hk3dfrz
6	j5le97h9hf2yq3h	...	...

## Exercise 4.3 (📌)

1. Pick one Security Question and explain how 💪 it is against attacks.
2. What would you recommend to pick as an answer? Assume that the risk of compromise is full takeover of your user account.

**Security Question** ⚠️ This cannot be changed later!

Your eldest siblings middle name?  
Mother's maiden name?  
Mother's birth date? (MM/DD/YY)  
Father's birth date? (MM/DD/YY)  
Maternal grandmother's first name?  
Paternal grandmother's first name?  
Name of your favorite pet?  
Last name of dentist when you were a teenager? (Do not include 'Dr.')  
Your ZIP/postal code when you were a teenager?  
Company you first work for as an adult?

# Prevention

## User IDs

- Use case insensitive and unique usernames/userids
- If using Email addresses as usernames, ensure [RFC 5321](#) validity
  - i. Check for presence of at least one `@` symbol in the address
  - ii. Ensure the local-part is no longer than 64 octets
  - iii. Ensure the domain is no longer than 255 octets
  - iv. Ensure the address is deliverable

**✗** *Do **not** try to invent your own RegEx to validate email addresses!*

## 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 Password Recovery Mechanism

1. Gather Identity Data or Security Questions
2. Verify Security Questions
3. Lock account immediately
4. Send a Token Over a Side-Channel
5. Allow user to change password in the existing session
6. Logging

## Secure Password Storage

- Do not limit character set and set long max lengths
- Use cryptographically strong credential-specific salt
- **Impose infeasible verification on attacker**
  - Adaptive one-way function ([Argon2](#), PBKDF2, bcrypt or scrypt)
  - Keyed functions (e.g. HMAC)
- Design password storage assuming eventual compromise
- **Upgrading your existing password hashing solution**

## Design for Failure

Having detected theft, a credential storage scheme must support continued operation by marking credential data as compromised:

1. Invalidate authentication shortcuts (e.g. login only with 2FA)
2. Disallow changes to security settings of user accounts
3. Load a new, stronger credential protection scheme
4. Set tainted / compromised bit until user resets credentials
5. Prompt for credential change & conduct out-of-band confirmation
6. Convert stored credentials to new scheme as user successfully log in

## 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 [<sup>1</sup>]




## 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, including memorizing and filling-in, generating random passwords on different sites etc. [<sup>2</sup>]

 KeePass		
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) [<sup>1</sup>]



## 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 user account (⭐⭐⭐⭐)

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

## Exercise 4.5 (🏠)



### Secure Quick Reliable Login (SQRL)

1. Read <https://www.grc.com/sqrl/sqrl.htm> and <http://sqrl.pl/guide> to learn how SQRL works
2. Prepare a convincing "sales pitch" (max. 5min) to convince your classmates and co-workers to use SQRL for secure authentication

## 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!*