Broken access control

AP HOGESCHOOL ANTWERPEN

AP.BE

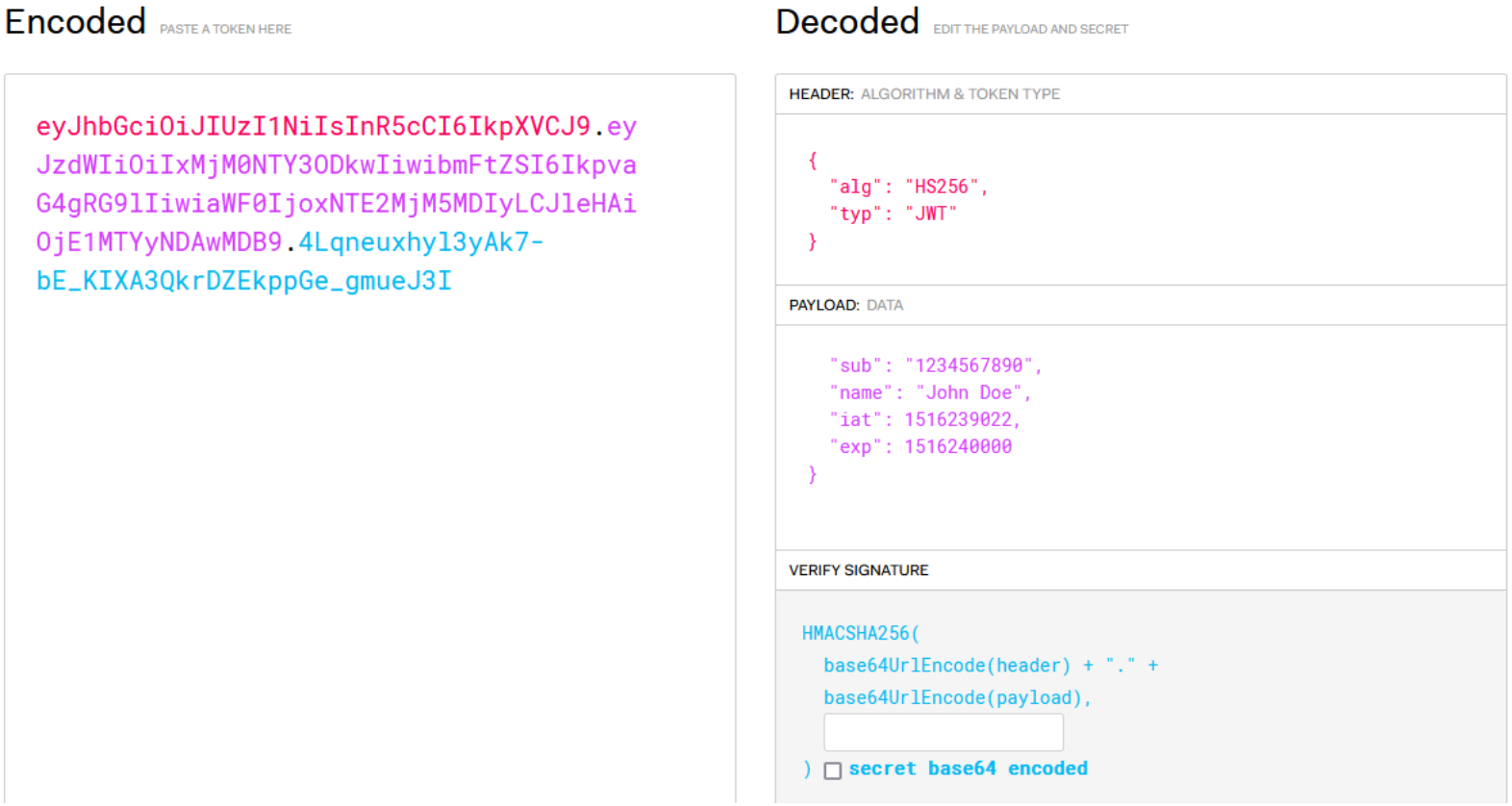
Token vulnerabilities

AP HOGESCHOOL ANTWERPEN

AP.BE

BROKEN ACCESS CONTROL: JWT TOKENS

How do vulnerabilities with (self-contained) JWT tokens arise?



BROKEN ACCESS CONTROL: JWT TOKENS

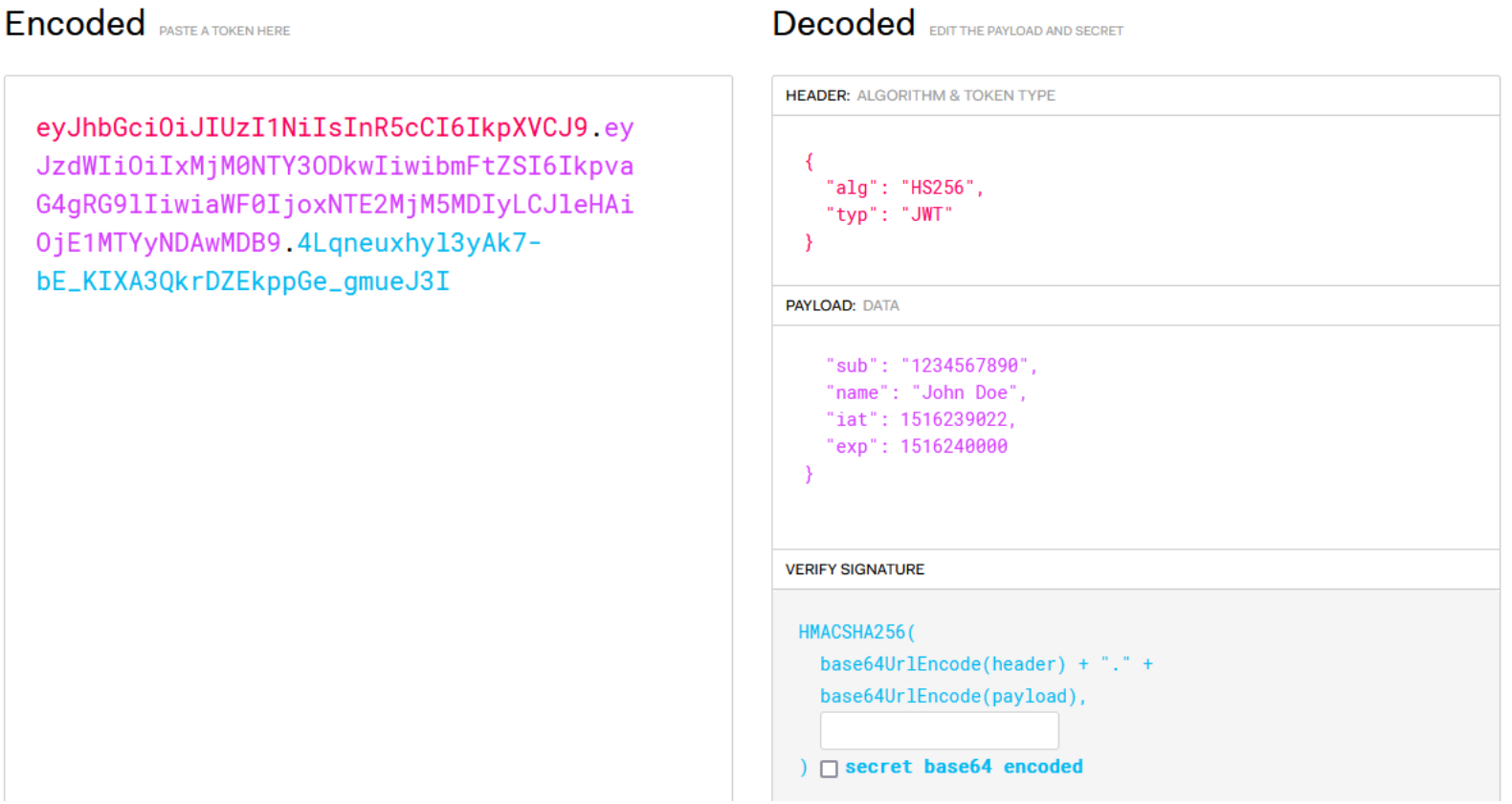
How do vulnerabilities with (self-contained) JWT tokens arise?

- Cannot be revoked → risk for compromise
- Insufficient signature validation:
  - Accepting tokens with no signature
  - Accepting arbitrary signatures
- Brute forcing secret key
- JWT header parameter injection

BROKEN ACCESS CONTROL: JWT TOKENS

Can typically not be revoked: server contains no state

→ long expiry means compromised token gives long unrestricted access



BROKEN ACCESS CONTROL: JWT TOKENS

Insufficient signature validation: accepting arbitrary signatures

- Straight up forgetting to validate header
- Accidentally doing ‘decode()’ instead of ‘verify()’

BROKEN ACCESS CONTROL: JWT TOKENS

Insufficient signature validation: Accepting tokens with no signature

{

…

“alg”: “none” → unsecured JWT,

…

}

Means the server will accept any claim, as it cannot be validated

BROKEN ACCESS CONTROL: JWT TOKENS

Brute forcing secret key:

Developers might forget to change test / default placeholder key

↓

Crackable using e.g. hashcat and wordlist of well-known secrets

(hashcat creates a signature of JWT and compares with given signature)

DEMO: CRACKING JWT KEY

- Using hashcat, we can bruteforce or use wordlists of common secret keys

BROKEN ACCESS CONTROL: JWT TOKENS

JWT Header injection:

- *jwk* (JSON web key): embedded public key in JWT
- *jku* (JSON web key set URL): URL from which server can fetch public key
- *kid* (Key ID): which key to choose in case there are multiple

BROKEN ACCESS CONTROL: JWT TOKENS

JWT Header injection:

- *jwk:*

- *jku:*
- *kid:*

BROKEN ACCESS CONTROL: JWT TOKENS

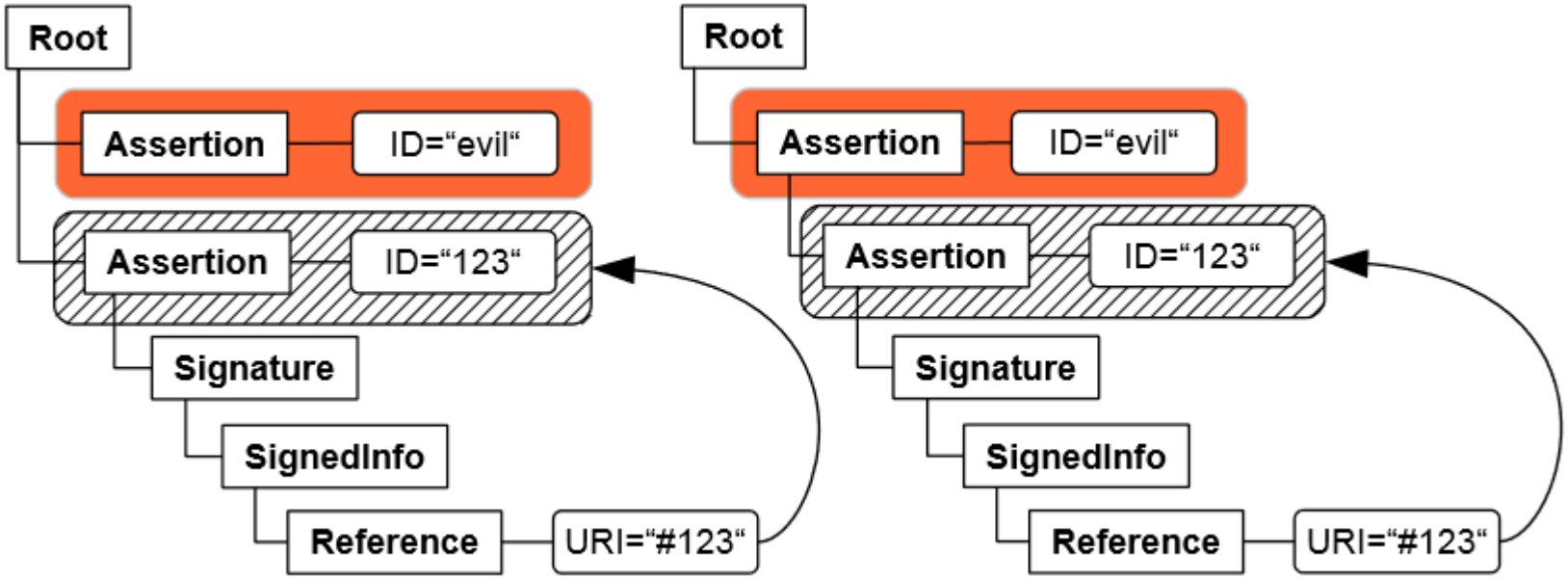
JWT Header injection:

- *jwk:* should only accept whitelisted keys, otherwise one can sign with an arbitrary key
- *jku:* similarly, should only accept trusted domains
- *kid:* No standard, sometimes points to a file. In case of symmetric files, could point to e.g. /dev/null (equal to signing the header with empty string)

BROKEN ACCESS CONTROL: SAML TOKENS (ASSERTIONS)

Common vulnerabilities:

- Only validating signature when present
- XML Signature Wrapping (XSW) attack:



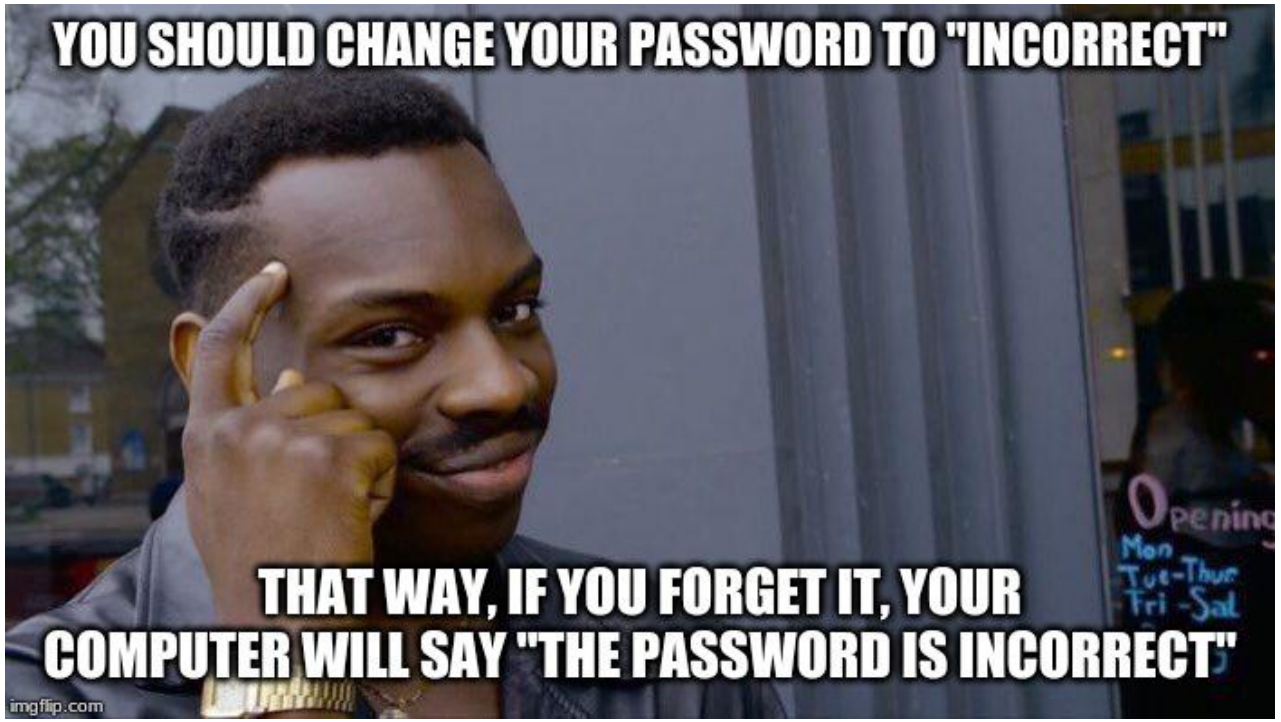
Passwords

AP HOGESCHOOL ANTWERPEN

AP.BE

AUTHENTICATION

- Three potential methods
  - Something you know



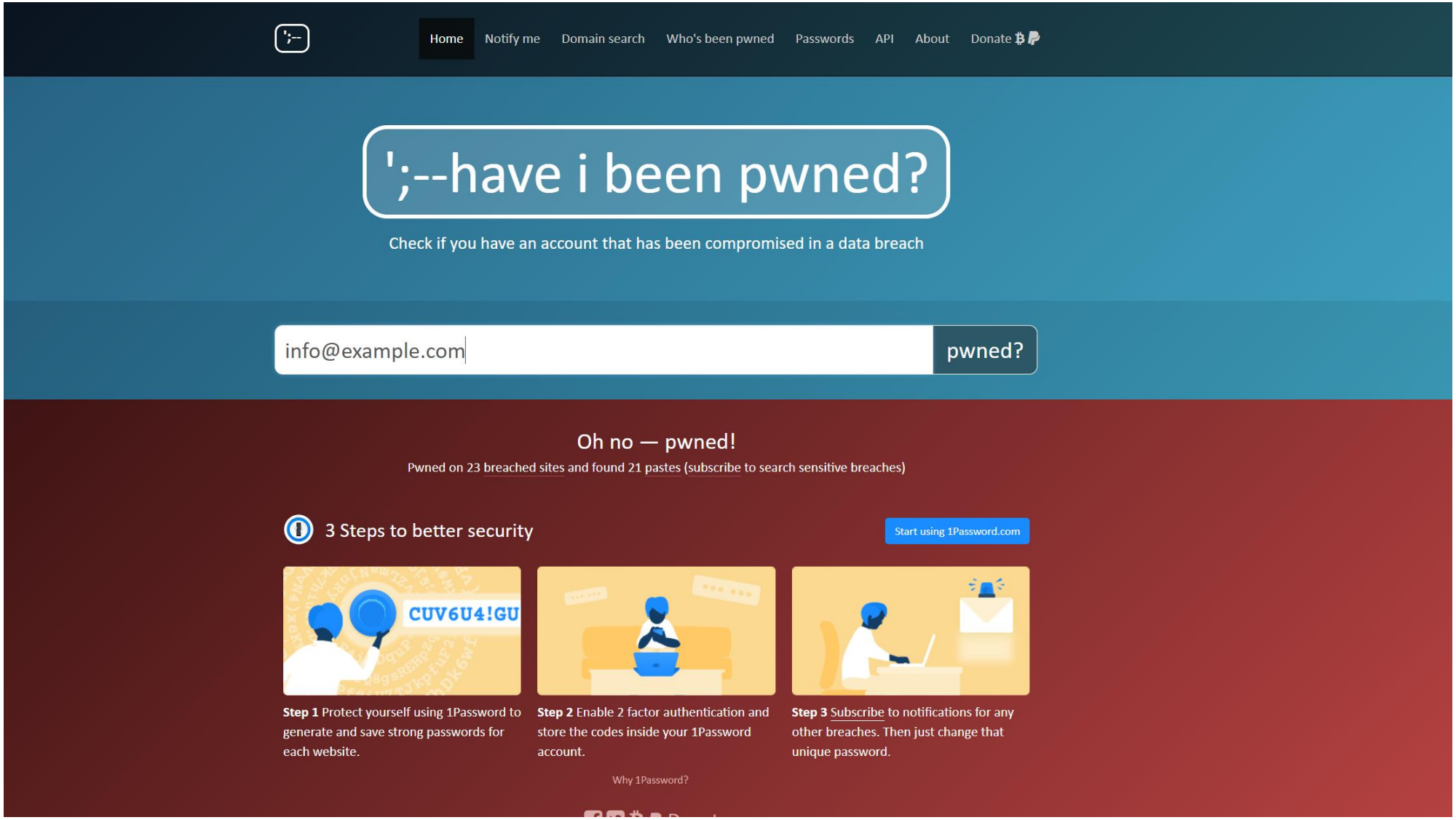
AUTHENTICATION

- Three potential methods
  - Something you know
  - Something you have

AUTHENTICATION

- Three potential methods
  - Something you know
  - Something you have
  - Something you are

SOMETHING YOU KNOW: PASSWORD



PASSWORDS

What do you do if you forget your password?

Forcing complex passwords leads to frustration for the user

What's your password?

PASSWORD RULES

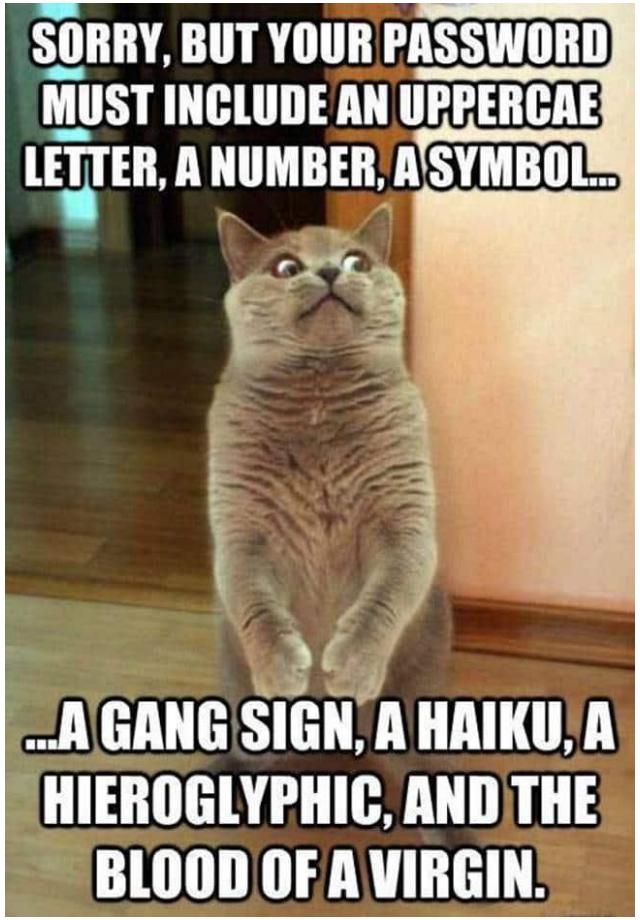
PASSWORD RULES

PASSWORD RULES

- Lists of known passwords
  - NordPass
  - SplashData
  - Keeper
  - National Cyber Security Centre
  - https://en.wikipedia.org/wiki/Wikipedia:10,000_most_common_passwords

PASSWORD RULES

PASSWORD RULES



PASSWORD RULES

- ✓ shall be at least 8 characters in length (if chosen by subscriber)
- ✓ do not store ‘hints’
- ✓ check passwords against known lists
- × no more periodic changes
- × no more complexity requirements

NIST Special Publication 800-63-3
Digital Identity Guidelines

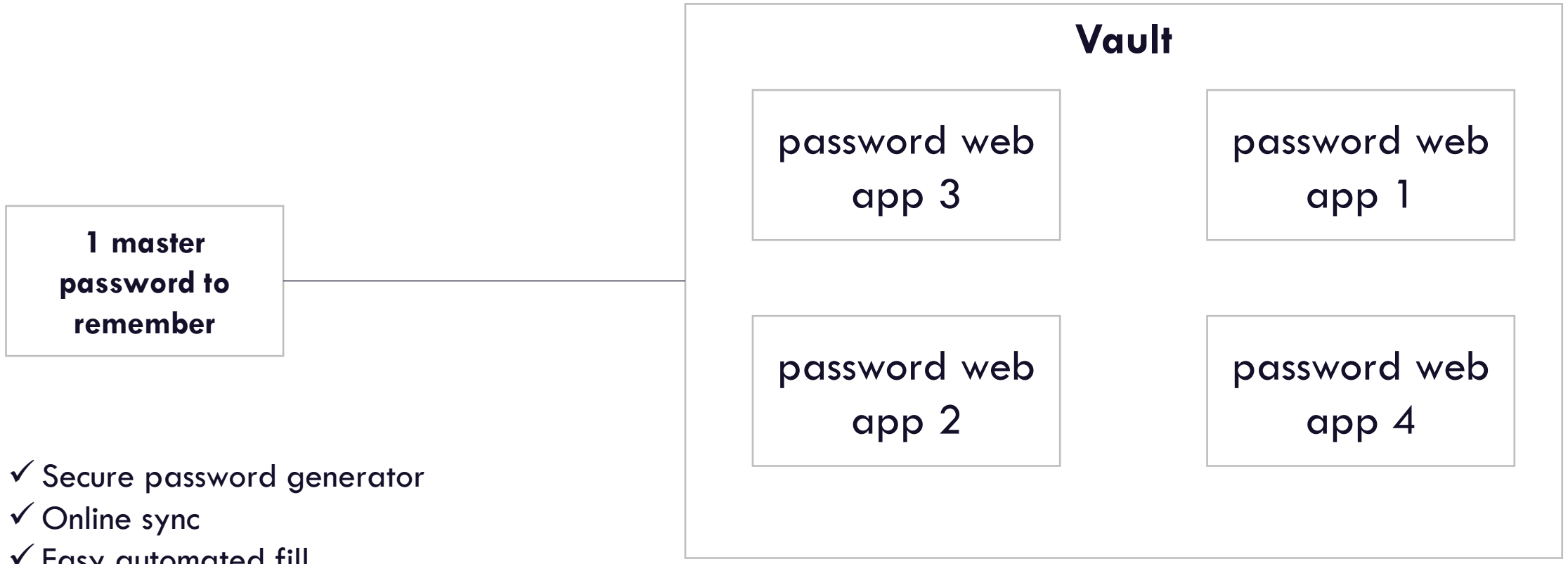
PASSWORD

For example, a user that might have chosen “password” as their password would be relatively likely to choose “Password1” if required to include an uppercase letter and a number, or “Password1!” if a symbol is also required.

Users also express frustration when attempts to create complex passwords are rejected by online services.

Highly complex memorized secrets introduce a new potential vulnerability: they are less likely to be memorable, and it is more likely that they will be written down or stored electronically in an unsafe manner.

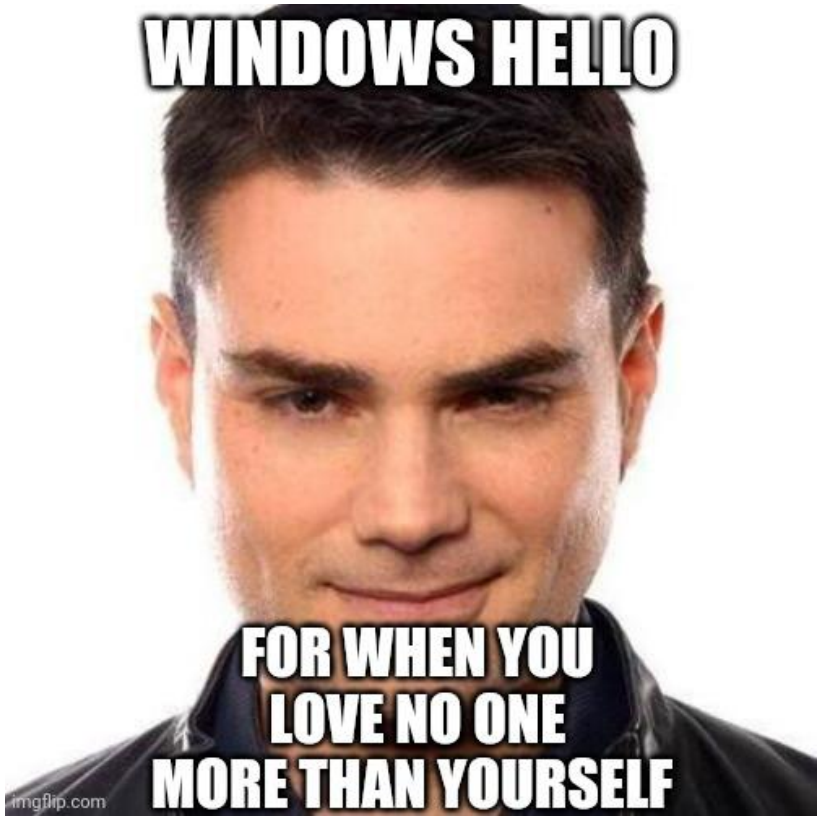
PASSWORD MANAGER



- ✓ Secure password generator
- ✓ Online sync
- ✓ Easy automated fill
- ✓ Secure storage

SOMETHING YOU HAVE

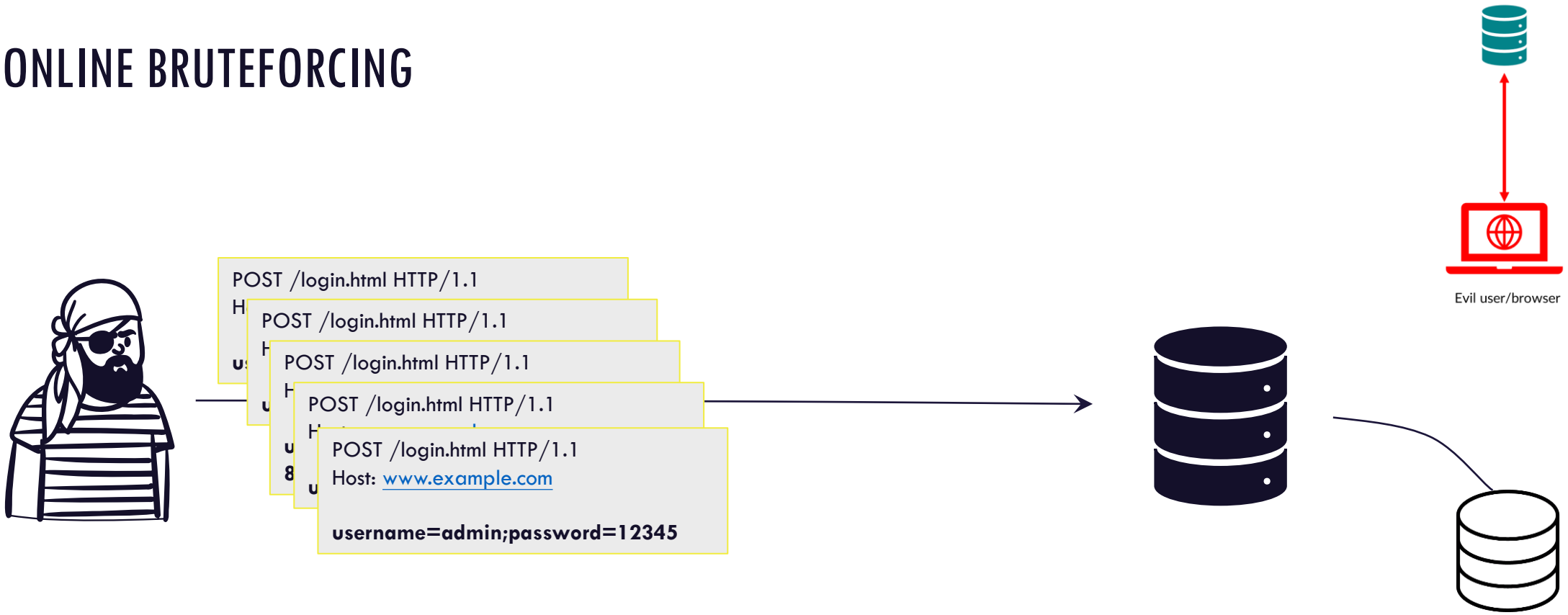
SOMETHING YOU ARE



MULTI FACTOR AUTHENTICATION

- Combination of 2 or more methods of authentication
- Makes pretending more difficult

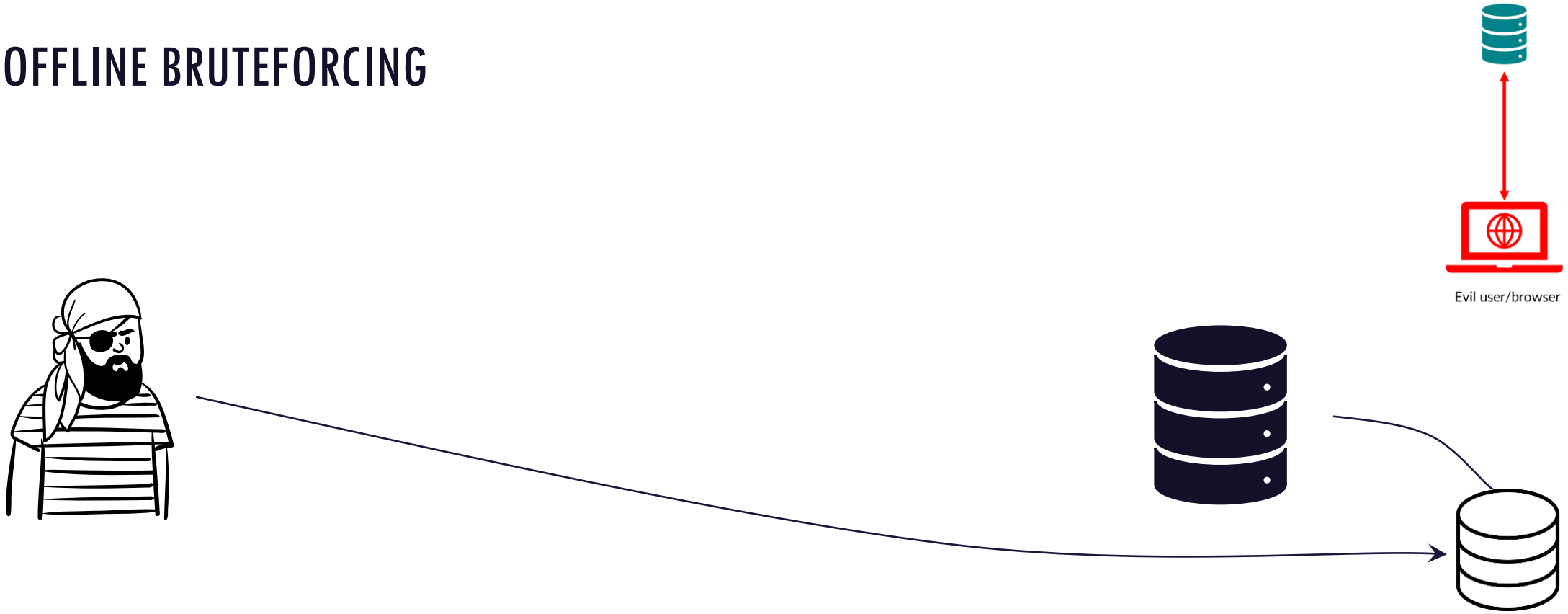
ONLINE BRUTEFORCING



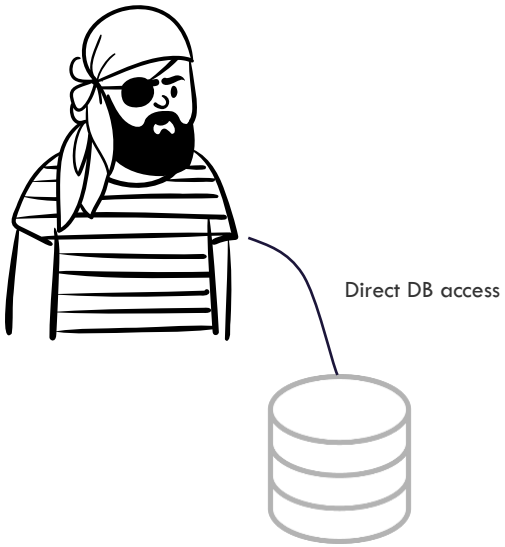
- ✓ Log failed login attempts and send them to your SIEM
- ✓ Ask for a captcha after X number of failed attempts
- ✓ Block account after X + Y number of failed attempts
- ✓ Simply use a good password

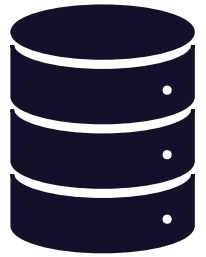
Fun to try: http://www.deathbycaptcha.com/user/login

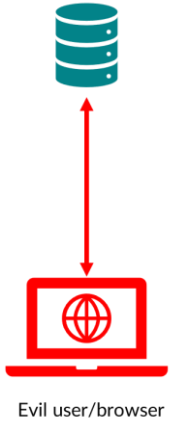
OFFLINE BRUTEFORCING



OFFLINE BRUTEFORCING







~~× Log failed login attempts and send them to your SIEM~~

~~× Ask for a captcha after X number of failed attempts~~

~~× Block account after X + Y number of failed attempts~~

✓Secure Password Storage

PASSWORD NIST GUIDELINES

SP 800-63A
Enrollment &
identity proofing

SP 800-63B
Authentication
and lifecycle mgmt

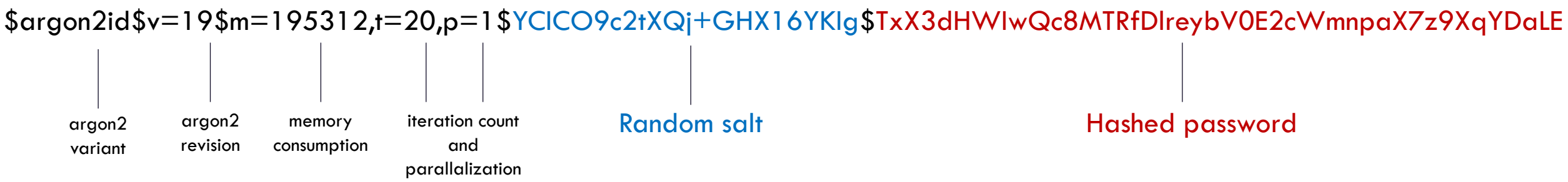
SP 800-63C
Federation and
assertions

PASSWORD STORAGE

SECURE PASSWORD STORAGE: SUMMARY

- Plain-text: …
  - bad
- (Cryptographic) Hash: e.g. SHA256
  - still bad: rainbow tables
- Hash + salt: e.g. SHA256 + CSPRNG
  - still bad: bruteforcing
- Slow hash + salt: e.g. argon2 (includes logic to generate salt)
  - standard

SECURE PASSWORD STORAGE: ARGON2



DEMO

- Demo cracking passwords with hashcat