TERMINOLOGIES

Authentication identifies the user and confirms that they say who they say they are.

- HTML form-based authentication
- Multi-factor mechanisms
- Windows-integrated authentication using NTLM or kerberos
  Authentication Vulnerabilities arise from insecure implementation of the authentication mechanisms in an application.

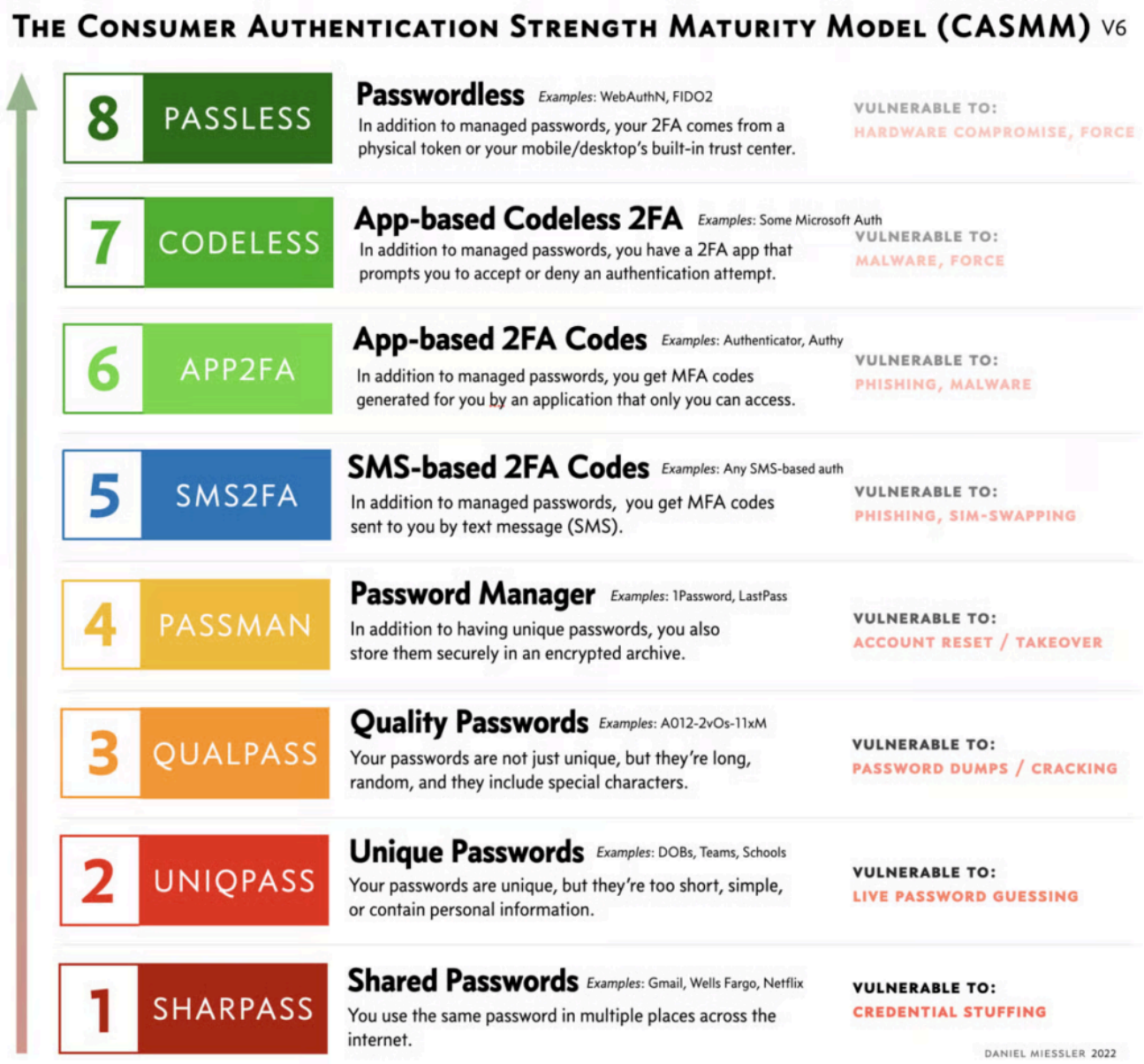
WHAT ARE AUTHENTICATION VULNERABILITIES

WEAK PASSWORD REQUIREMENTS

Having no or minimal controls over the quality of users passwords.

- Very short or blank
- Common dictionary words or names
- Password is the same as the username
- Use of default password

- Missing or ineffective MFA



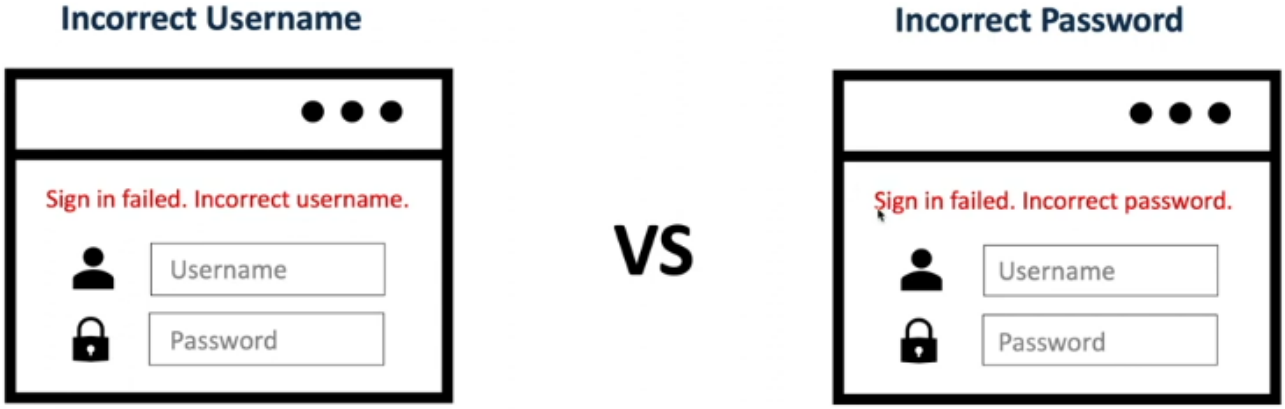
IMPROPER RESTRICTION OF AUTHENTICATION ATTEMPTS

Application permits brute force or other automated attacks.

- Login page
- OTP / MFA page
- Change password page

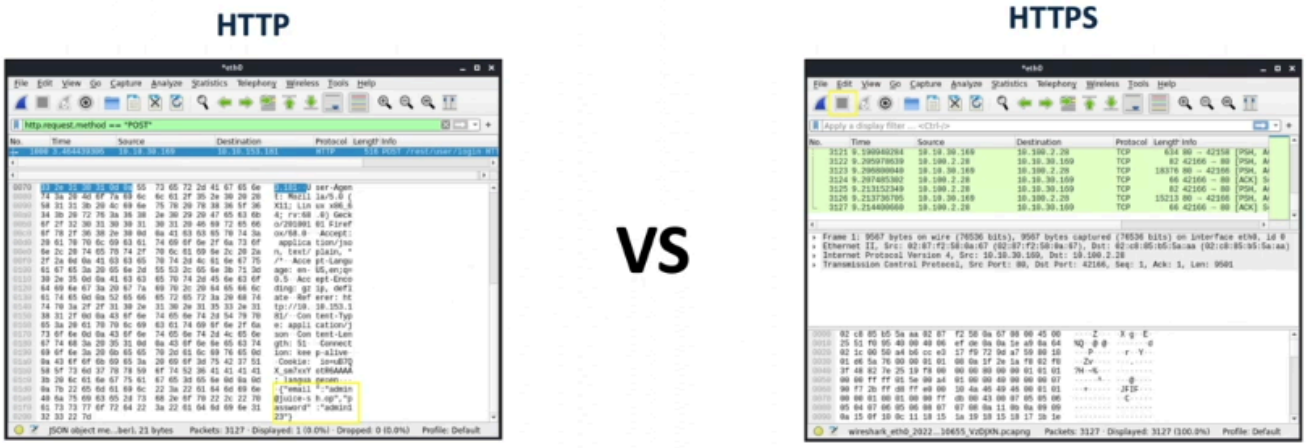
VERBOSE ERROR MESSAGE

The application outputs a verbose error message that allows for username enumeration.



VULNERABLE TRANSMISSION OF CREDENTIALS

The application uses an encrypted HTTP connection to transmit login credentials



INSECURE FORGET PASSWORD FUNCTIONALITY

Design weaknesses in the forgotten password functionality usually make the weakest link that can be used to attack the applications overall authentication logic.

DEFECTS IN MULTISTAGE LOGIN MECHANISM

Insecure implementation of the MFA function

REQUEST 1

```
POST /login-steps/first HTTP/1.1
Host: vulnerable-website.com
...

username=carlos&password=qwerty
```

→

REQUEST 2

```
POST /login-steps/second HTTP/1.1
Host: vuln-website.com Cookie:
account=carlos
...
verification-code=123456
```

Exploiting:

Change the "account" cookie to the victims username and compromise the victims account.

INSECURE STORAGE OF CREDENTIALS

Use plain text, encrypted or weakly hashed password data stores.

	Algorithm	Password
✗	None	Password1!
✗	AES256 and B64	jc2ZRviEVUuLV7Ljc2q7YQ==
✗	MD5	0cef1fb10f60529028a71f58e54ed07b
✗	SHA256	1D707811988069CA760826861D6D63A10E8C3B7F171C4441A6472EA58C11711B

HOW TO FIND AND EXPLOIT AUTHENTICATION FLAWS ?

WEAK PASSWORD COMPLEXITY REQUIREMENTS

- Review the website for any description of the rules
- If self registration is possible, attempt to register several accounts with different kinds of weak passwords to discover what rules are in place
  - Very short or blank.
  - Common dictionary words or names.
  - Password is the same as the username.
- If you control a single account and password change is possible, attempt to change the password to various weak values.

IMPROPER RESTRICTION OF AUTHENTICATION ATTEMPTS

- Manually submit several bad login attempts for an account you control
- After 10 failed login attempts, if the application does not return a message about account locked, attempt to log in correctly. If it works, then there is no lockout mechanism.
  - Run a brute force attack to enumerate the valid password. Tools: Hydra, Burp Intruder, etc.
- If the account is locked out, monitor the requests and responses to determine if the lockout mechanism is insecure.

NOTE: Apply this test on all authentication pages.

VERBOSE ERROR MESSAGE

- Submit a request with a valid username and an invalid password
- Submit a request with an invalid username
- Review both responses for any differences in the status code, any redirects, information displayed on the screen, HTML page source, or even the time to process the request.
- If there is a difference, run a brute force attack to enumerate the list of valid username in the application

NOTE: Apply this test on all authentication pages.

VULNERABLE TRANSMISSION OF CREDENTIALS

- Perform a successful login while monitoring all traffic in both directions between the client and server
- Look for instances where credentials are submitted in a URL query string or as a cookie, or are transmitted back from the server to the client.
- Attempt to access the application over HTTP and if there are any redirections to HTTPS.

INSECURE FORGOT PASSWORD FUNCTIONALITY

- Identify if the application has any forgotten password functionality.
- If it does, perform a complete walk-through of the forgot password functionality using an account you have control while intercepting the requests / responses in a proxy.
- Review the functionality to determine if it allows for username enumeration or brute-force attacks.
- If the application generates an email containing a recovery URL, obtain a number of these URLs and attempt to identify any predictable patterns or sensitive information included in the URL. Also check if the URL is long lived and does not expire.

DEFECTS IN MULTISTAGE LOGIN MECHANISM

- Identify if the application uses a multistage login mechanism
- if it does, perform a complete walk-through using an account you have control of while intercepting the requests / responses in a proxy.
- Review the functionality to determine if it allows for a username enumeration or brute-force attacks.

INSECURE STORAGE OF CREDENTIALS

- Review all the applications authentication related functionality. If you find any instances where the users password is transmitted to the client plaintext or obfuscated, this indicates the passwords are being stored insecurely.
- If you gain remote code execution(RCE) on the server, review the database to determine if the passwords are stored insecurely.
- Conduct technical interviews with the developers to review how passwords are stored in the backend database.

PREVENTING AUTHENTICATION VULNERABILITIES

- Wherever possible, implement multi-factor authentication.
- Change all default credentials.
- Always use an encrypted channel / connection (HTTPS) when sending user credentials.
- Only POST requests should be used to transmit credentials to the server.
- Stored credentials should be hashed and salted using cryptographically secure algorithms.
- Use identical, generic error messages on the login form when the user enters incorrect credentials.
- Implement an effective password policy that is compliant with NIST 800-63-b's guidelines.
    - Use a simple password checker to provide real time feedback on the strength of the password. For example: zxcvbn JavaScript library.
- Implement robust brute force protection on all authentication pages.
- Audit any verification or validation logic thoroughly to eliminate flaws.