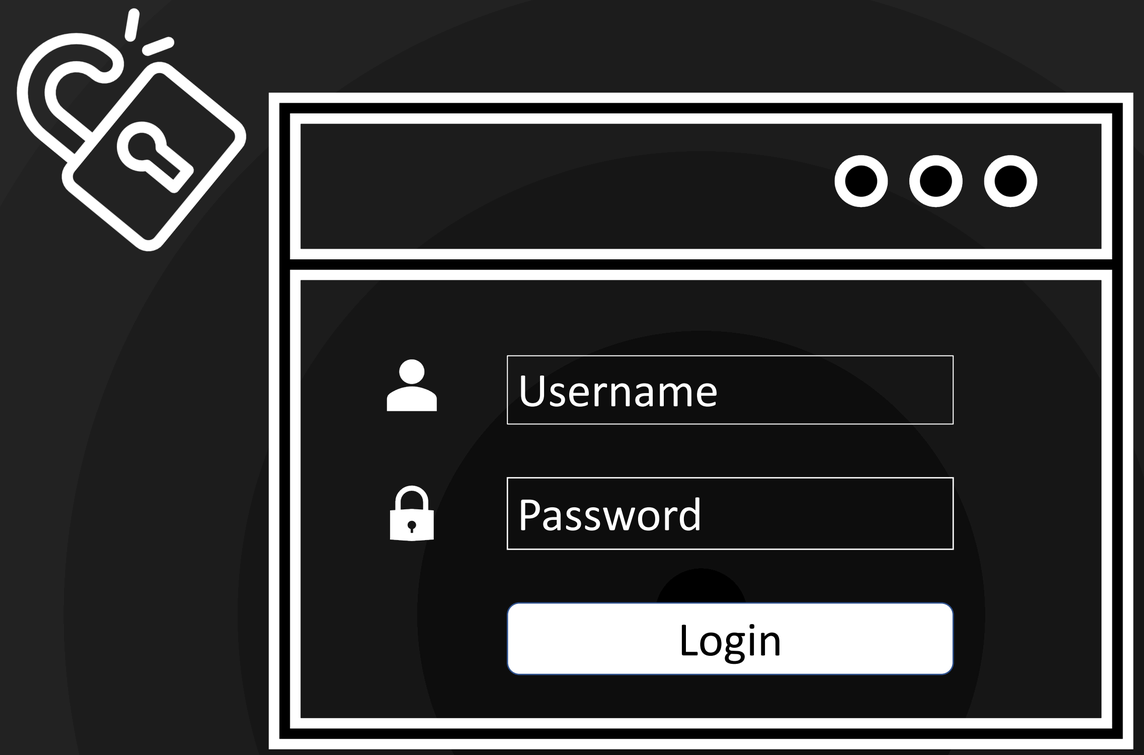


Authentication Vulnerabilities

Author: Rana Khalil (@rana__khalil)

Agenda

WHAT ARE AUTHENTICATION FLAWS?

HOW DO YOU FIND AND EXPLOIT THEM?

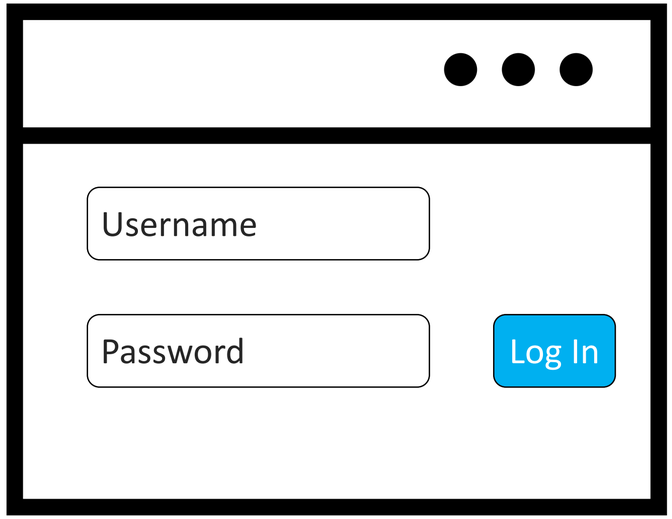
HOW DO YOU PREVENT THEM?

WHAT ARE AUTHENTICATION VULNERABILITIES?

Some Terminology...

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
- ...

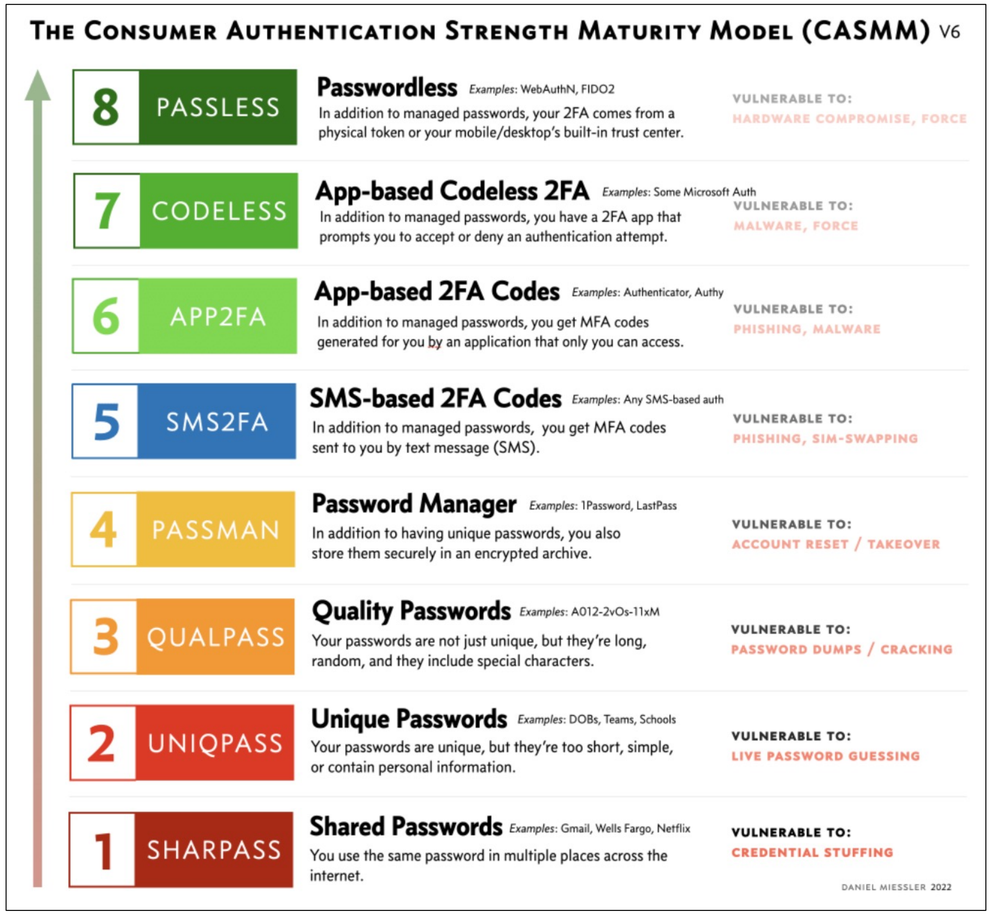


Authentication Vulnerabilities arise from insecure implementation of the authentication mechanisms in an application.

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



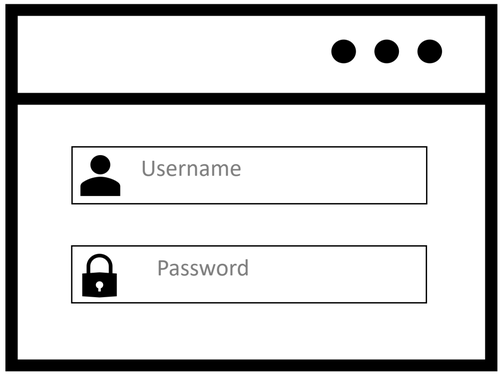
Link: https://danielmiessler.com/blog/casmm-consumer-authentication-security-maturity-model/

Improper Restriction of Authentication Attempts

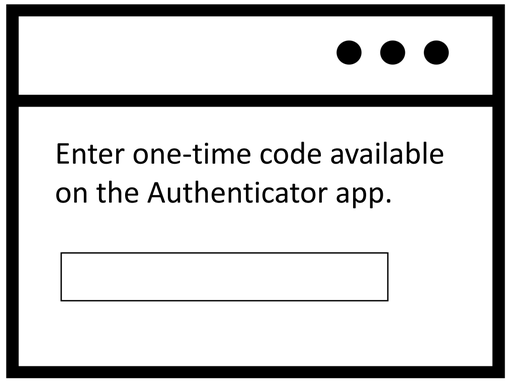
Application permits brute force or other automated attacks.

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

Login Page



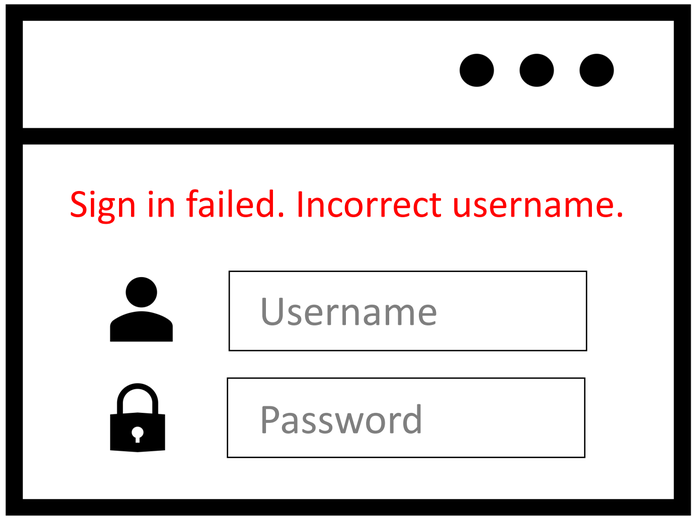
OTP Page



Verbose Error Message

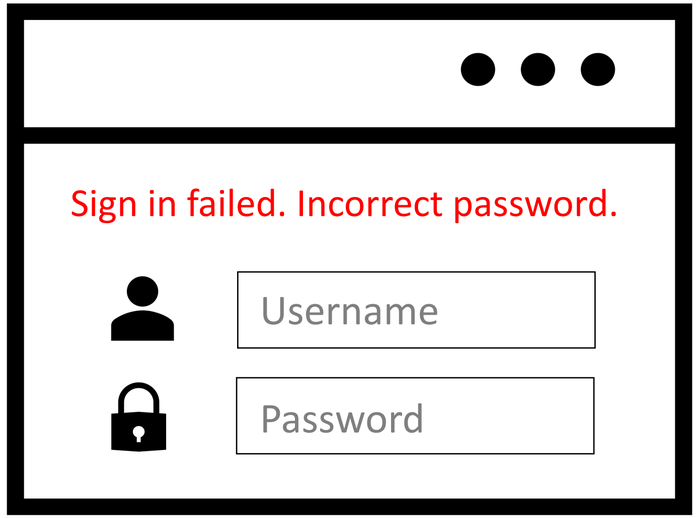
The application outputs a verbose error message that allows for username enumeration.

Incorrect Username



VS

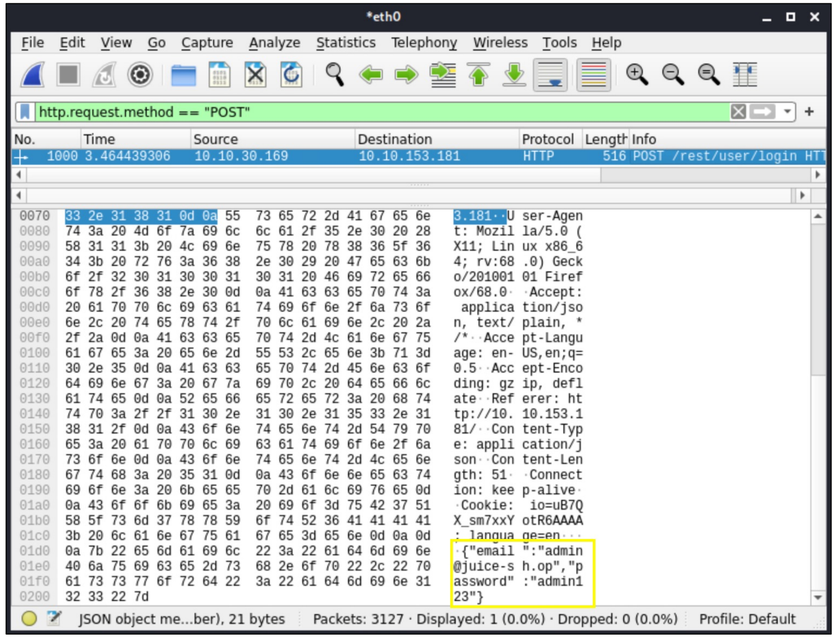
Incorrect Password



Vulnerable Transmission of Credentials

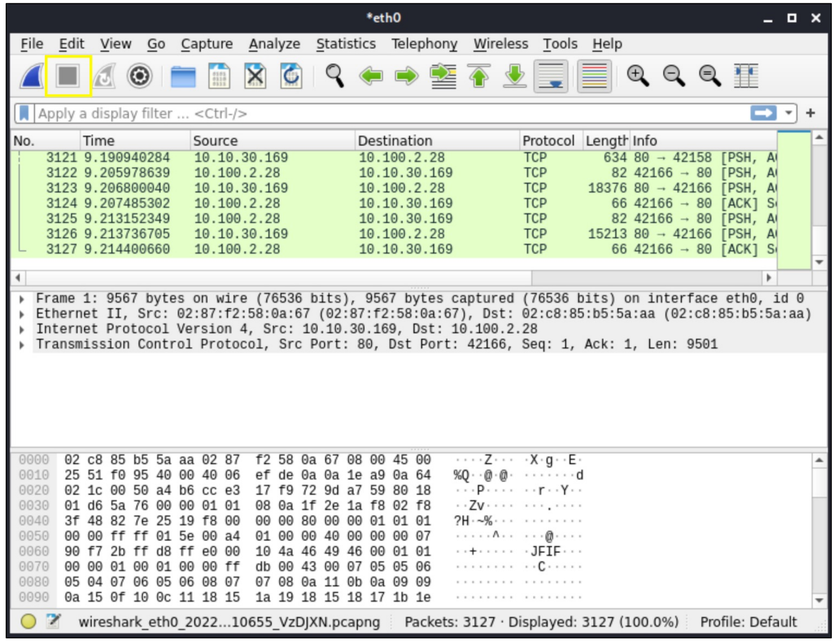
The application uses an unencrypted HTTP connection to transmit login credentials.

HTTP



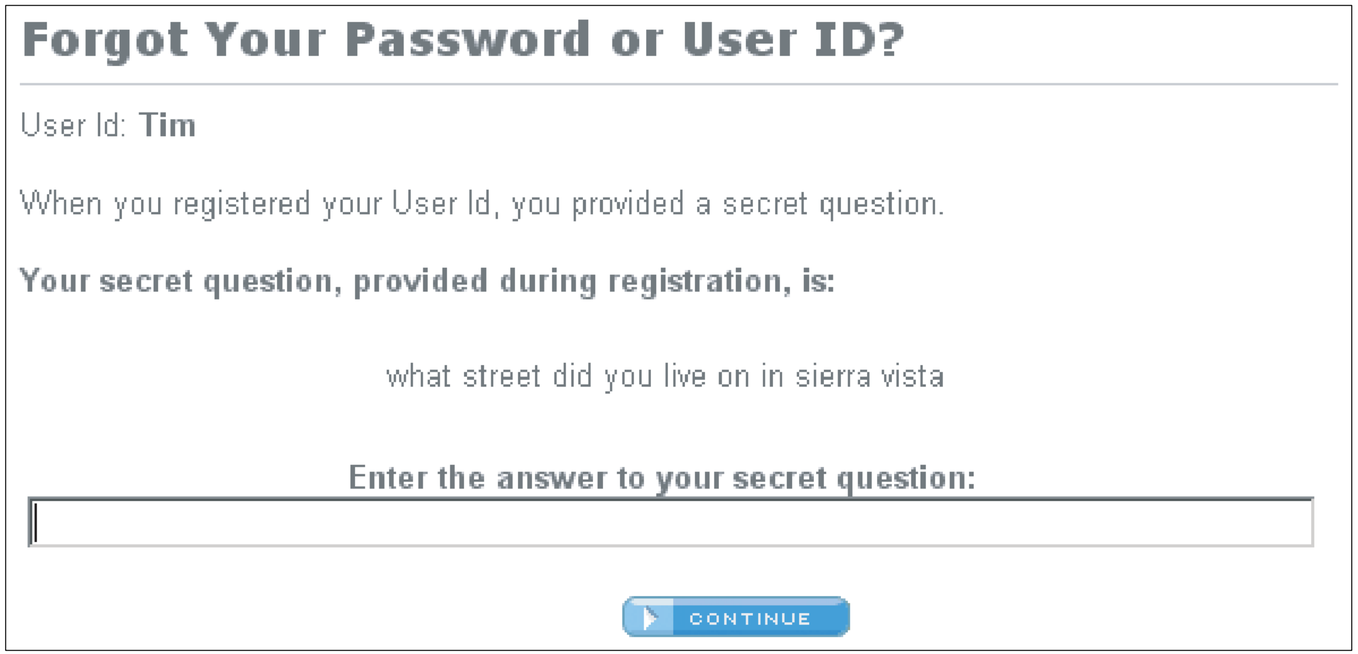
VS

HTTPS



Insecure Forgot Password Functionality

Design weaknesses in the forgotten password functionality usually make the weakest link that can be used to attack the application's 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
```

How can this be exploited?

Change the “account” cookie to the victim’s username and compromise the victim’s account.

Insecure Storage of Credentials

Uses plain text, encrypted, or weekly hashed password data stores.

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

Impact of Authentication Vulnerabilities

- Unauthorized access to the application.
  - **C**onfidentiality – Access to other users' data.
  - **I**ntegrity – Access to update other users' data
  - **A**vailability – Access to delete users and their data.
- Can sometimes be chained with other vulnerabilities to gain remote code execution on the host operating system.

OWASP Top 10

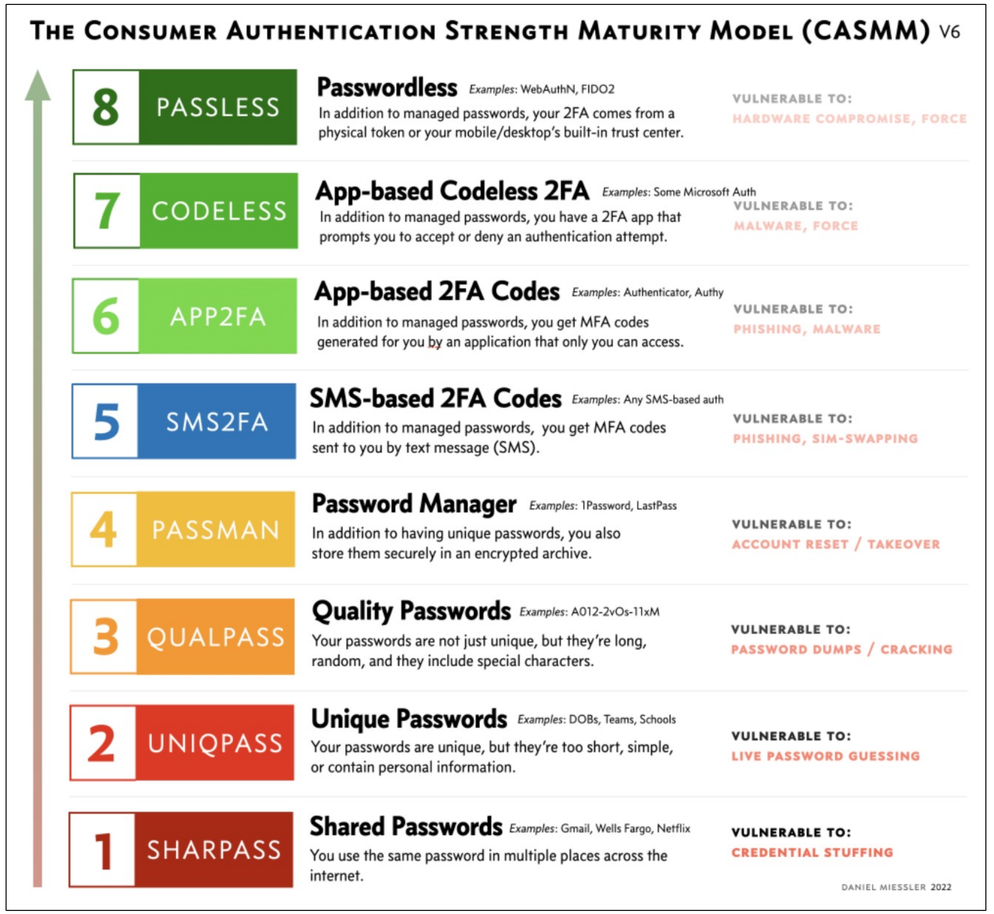
OWASP
Open Web Application Security Project

OWASP Top 10 - 2013	OWASP Top 10 - 2017	OWASP Top 10 - 2021
A1 – Injection	A1 – Injection	A1 – Broken Access Control
A2 – Broken Authentication and Session Management	A2 – Broken Authentication	A2 – Cryptographic Failures
A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure	A3 - Injection
A4 – Insecure Direct Object References	A4 – XML External Entities (XXE)	A4 – Insecure Design
A5 – Security Misconfiguration	A5 – Broken Access Control	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Security Misconfiguration	A6 – Vulnerable and Outdated Components
A7 – Missing Function Level Access Control	A7 – Cross-Site Scripting (XSS)	A7 – Identification and Authentication Failures
A8 – Cross-Site Request Forgery (CSRF)	A8 – Insecure Deserialization	A8 – Software and Data Integrity Failures
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities	A9 – Security Logging and Monitoring Failures
A10 – Unvalidated Redirects and Forwards	A10 – Insufficient Logging & Monitoring	A10 – Server-Side Request Forgery (SSRF)

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.



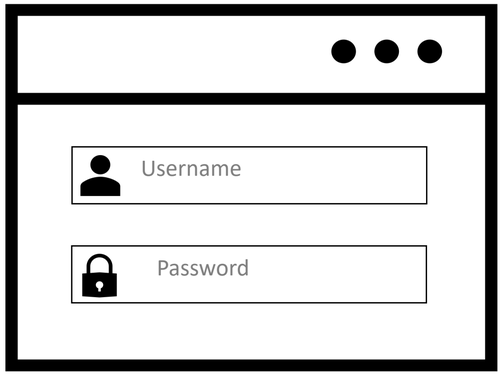
Link: https://danielmiessler.com/blog/casmm-consumer-authentication-security-maturity-model/

Improper Restriction of Authentication Attempts

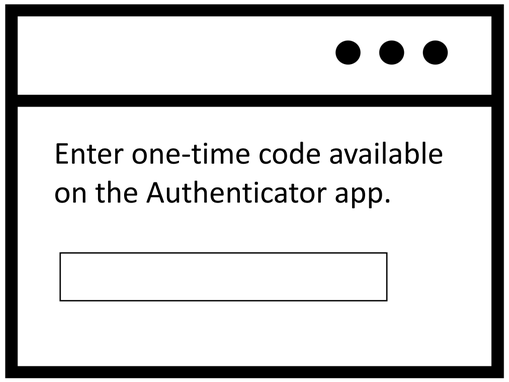
- Manually submit several bad login attempts for an account you control.
- After 10 failed login attempts, if the application does not return a message about account lockout, attempt to log in correctly. If it works, then there is no lockout mechanism.
  - Run a brute force attack to enumerate the valid 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.

Login Page



OTP Page



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 usernames in the application.

NOTE: Apply this test on all authentication pages.

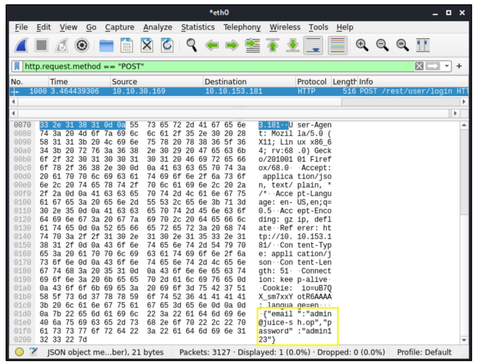
Incorrect Username

Incorrect Password

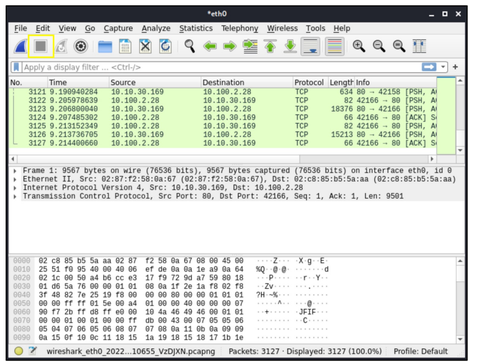
Vulnerable Transmission of Credentials

- 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.

HTTP



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 of 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 username enumeration or brute-force attacks.

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
```

Insecure Storage of Credentials

- Review all the application's authentication related functionality. If you find any instances where the user's 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.

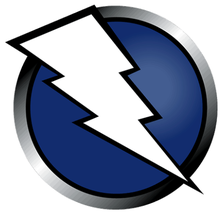
	Algorithm	**Password**
✗	None	Password1!
✗	AES256 and B64	jc2ZRviEVUuLV7Ljc2q7YQ==
✗	MD5	0cef1fb10f60529028a71f58e54ed07b
✗	SHA256	1D707811988069CA760826861D6D63A10E8C3B7F171C4441A6472EA58C11711B

Exploiting Authentication Flaws Labs

- LAB | APPRENTICE | Username enumeration via different responses »
- LAB | APPRENTICE | 2FA simple bypass »
- LAB | APPRENTICE | Password reset broken logic »
- LAB | PRACTITIONER | Username enumeration via subtly different responses »
- LAB | PRACTITIONER | Username enumeration via response timing »
- LAB | PRACTITIONER | Broken brute-force protection, IP block »
- LAB | PRACTITIONER | Username enumeration via account lock »
- LAB | PRACTITIONER | 2FA broken logic »
- LAB | PRACTITIONER | Brute-forcing a stay-logged-in cookie »
- LAB | PRACTITIONER | Offline password cracking »
- LAB | PRACTITIONER | Password reset poisoning via middleware »
- LAB | PRACTITIONER | Password brute-force via password change »
- LAB | EXPERT | Broken brute-force protection, multiple credentials per request »
- LAB | EXPERT | 2FA bypass using a brute-force attack »

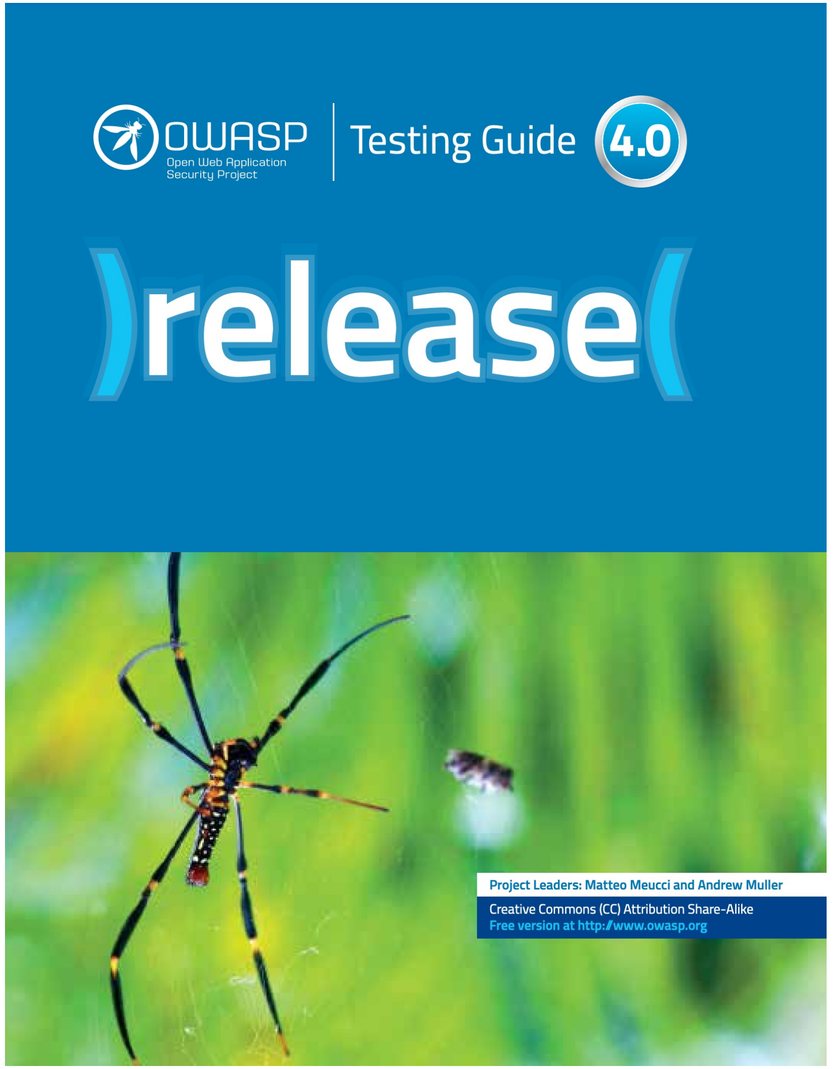
Automated Exploitation Tools

Web Application Vulnerability Scanners (WAVS)



Web Application Security Testing (WSTG) Guide

4.4 Authentication Testing



Testing Guide Link: : *https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf*

HOW TO PREVENT AUTHENTICATION VULNERABILITIES?

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.

Preventing Authentication Vulnerabilities

- …
- 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.

Application Security Verification Standard (ASVS)

V2: Authentication Verification Requirements

#	Description	L1	L2	L3	CWE	NIST §
2.4.1	Verify that passwords are stored in a form that is resistant to offline attacks. Passwords SHALL be salted and hashed using an approved one-way key derivation or password hashing function. Key derivation and password hashing functions take a password, a salt, and a cost factor as inputs when generating a password hash. (C6)		✓	✓	916	5.1.1.2
2.4.2	Verify that the salt is at least 32 bits in length and be chosen arbitrarily to minimize salt value collisions among stored hashes. For each credential, a unique salt value and the resulting hash SHALL be stored. (C6)		✓	✓	916	5.1.1.2
2.4.3	Verify that if PBKDF2 is used, the iteration count SHOULD be as large as verification server performance will allow, typically at least 100,000 iterations. (C6)		✓	✓	916	5.1.1.2
2.4.4	Verify that if bcrypt is used, the work factor SHOULD be as large as verification server performance will allow, typically at least 13. (C6)		✓	✓	916	5.1.1.2
2.4.5	Verify that an additional iteration of a key derivation function is performed, using a salt value that is secret and known only to the verifier. Generate the salt value using an approved random bit generator [SP 800-90Ar1] and provide at least the minimum security strength specified in the latest revision of SP 800-131A. The secret salt value SHALL be stored separately from the hashed passwords (e.g., in a specialized device like a hardware security module).		✓	✓	916	5.1.1.2

ASVS Guide Link: : *https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf*

Resources

- Web Security Academy – Authentication Vulnerabilities
  - *https://portswigger.net/web-security/authentication*
- Web Application Hacker's Handbook
  - *Chapter 6 – Attacking Authentication*
- OWASP Web Security Testing Guide – Authentication Testing
  - *https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/04-Authentication_Testing/README*
- OWASP Top 10 – A07 Identification and Authentication Failures
  - *https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/*
- OWASP Application Security Verification Standard – V2 Authentication Verification Requirements
  - *https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf*