# **Authentication vulnerabilities**

Authentication is the process of verifying that a user is who they claim to be. Authorization involves verifying whether a user is allowed to do something.

**Tools:**

<https://www.browserling.com/tools/all-hashes>

Reverse lookup of md5sum hash

<https://md5.gromweb.com/?md5=26323c16d5f4dabff3bb136f2460a943>

Three authentication factors

* Something you know, such as a password or the answer to a security question. These are sometimes referred to as "knowledge factors".
* Something you have, that is, a physical object like a mobile phone or security token. These are sometimes referred to as "possession factors".
* Something you are or do, for example, your biometrics or patterns of behavior. These are sometimes referred to as "inherence factors".

## Vulnerabilities in authentication mechanisms

2 FA can be brute forced if the logic is flaws

* Check if IP block happens
* Check if IP block is unblocked after successful login so use your own accouunt to do login and hence unblock happens then continue the attack for the victim user
* Sometimes the 2nd FA is not fully secure because the first FA is authenticated and then this fact is saved in cookie so attacker can swap this with victim user and then brute force it for the 2FA
* Sometimes users have too many attempts error when user exist in system
* Sometime all password can be passed in the json if the password and username is passed as json and password field is a list which is a valid json
* Sometime stay login triggers a store of cookie flag which are encoded with static values and can be used to bypass login step altogather
* Sometimes the reset token is not checked who has generated the token so your account token can be used to reset the password of victim
* While resetting the password, check if the reset password is sending username also to check whose password is being reset and also sometimes the API can accessed without login
* Also figure out different combination of error messages that you get and design the logic from it

To Bypass different IP

X-Forwarded-For: 1.2.3.4

HTTP header used to identify the originating IP address of a client connecting to a web server through an HTTP proxy or a load balancer

Its very finky where don’t pass the https axnd also without space before :

X-Forwarded-Host: exploit-0a1500cc041579c2801593f001e6008d.exploit-server.net

"X-Forwarded-Host" (XFH) header is an HTTP header that is used to pass the original host requested by the client to a web server, especially when requests pass through proxy servers or load balancers

Problem start from here: https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-poisoning-via-middleware

X-Forwarded-Host

https://portswigger.net/web-security/authentication/password-based

**Easy Level:**

BruteForcing the password and find username via email

Username enumeration: Username enumeration is when an attacker is able to observe changes in the website's behavior in order to identify whether a given username is valid.

while checking, things to look for change in:

* Status codes
* Error messages: could be different by a “space or stop so attack and do grep on error msg:
* Response times: by giving very long password
  + Columns of response time is on title bar of intruder attack
    - When username is correct and password is wrong and long, it might have different response time and if username is worng then it has same resposne time
    - If username and password are correct then status would be different

Complete the rest of lab after 3 labs

https://portswigger.net/web-security/authentication/password-based

# OAuth 2.0 [authentication vulnerabilities](https://portswigger.net/web-security/authentication)

It works by defining a series of interactions between three distinct parties, namely a **client application**, **a resource owner**, and the **OAuth service provider**.

**Grant types** also known as **OAuth flows**

egardless of which OAuth grant type is being used, the first request of the flow will always be a request to the /authorization endpoint containing a number of query parameters that are used specifically for OAuth. In particular, keep an eye out for the client\_id, redirect\_uri, and response\_type parameters. For example, an authorization request will usually look something like this:

Once you know the hostname of the authorization server, you should always try sending a GET request to the following standard endpoints:

* /.well-known/oauth-authorization-server
* /.well-known/openid-configuration

These will often return a JSON configuration file containing key information

**These parameters to auth is very important**

https://oauth-0ad100f503d8a6a38123f5ce029f0038.oauth-server.net/auth?client\_id=z41fnkmc3skprekrkm3mf&redirect\_uri=https://0a9b00530394a6ec811df73a00b700ac.web-security-academy.net/oauth-callback&response\_type=token&nonce=1767142462&scope=openid%20profile%20email