you have a website which allows you to upload private photos but you've discovered an IDOR which allows you to view any photo you wish. **Think deeper and think**, "If they aren't checking that I own the ID I'm querying, what else have they forgotten to do certain permission checks on?". If you can sign up as various different roles (admin, guest), can you perform admin actions as a guest? Can non paying members access paid features? IDORs are fun to find as sometimes you can discover the entire web app is broken.

As well as hunting for integer values I will also try simply injecting ID parameters. Anytime you see a request and the postdata is JSON, `{"example":"example"}`, try simply injecting a new parameter name, `{"example":"example","id":"1"}`. When the JSON is parsed server-side you literally have no idea how it may handle it, so why not try? This not only applies to JSON requests but **all** requests, but I typically have a higher success rate when it's a JSON payload. **(look for PUT requests!**)

CORS (Cross-Origin Resource Sharing)

Another really common area to look for **filtering** *(remember, I am interested in finding specific filters in place to try bypass!)* is when you see "Access-Control-Allow-Origin:" as a header on the response. You will also sometimes need "Access-Allow-Credentials:true" depending on the scenario. These headers allow for an external website to read the contents of the website. So for example if you had sensitive information on https://api.zseano.com/user/ and you saw "Access-Control-Allow-Origin: https://www.yoursite.com/" then you could read the contents of this website successfully via yoursite.com. Allow-credentials will be needed if session cookies are required on the request. Developers will create filters to only allow for **their domain** to read the contents but remember, when there is a filter there is usually a bypass! The **most common** approach to take is to try anythinghere**theirdomain.com** as sometimes they will only be checking for if their domain is found, which in this case it is, but we control the domain! When hunting for CORS misconfigurations you can simply add "Origin: theirdomain.com" onto every