doing checks on the imagesize. Sometimes if you leave the image header this is enough to bypass the checks.

```
------WebKitFormBoundaryoMZOWnpiPkiDc0yV
Content-Disposition: form-data; name="oauth_application[logo_image_file]";
filename="testing1.html"
Content-Type: text/html

‰PNG
<script>alert(0)</script>
```

File uploads will more than likely contain some type of filter to prevent malicious uploads so make sure to spend enough time testing them.

Insecure Direct Object Reference (IDOR)

An example of an IDOR bug is simply a url such as https://api.zseano.com/user/1 which when queried will give you the information to the user id “1”. Changing it to user id “2” should give you an error and refuse to show you other users' details, however if they are vulnerable, then it will allow you to view that users details. In a nutshell, IDOR is about changing integer values (numbers) to another and seeing what happens. That’s the “explain like i’m 5”.

Of course it isn’t always as simple as looking for just integer (1) values. Sometimes you will see a GUID (2b7498e3-9634-4667-b9ce-a8e81428641e) or another type of encrypted value. Brute forcing GUIDs is usually a dead-end so at this stage I will check for any leaks of this value. I once had a bug where I could remove anyone's photo but I could not enumerate the GUID values. Visiting a users’ public profile and viewing the source revealed that the users photo GUID was saved with the file name (https://www.example.com/images/users/2b7498e3-9634-4667-b9ce-a8e81428641e/photo.png).