you can simply change that to £10,000 and bypass their limit then you've done nothing but take advantage of the feature right in front of you. No scanning, no weird filters, no hacking involved really. Just simply checking if the process works how it should work.

One common area I look for when hunting for application logic bugs is **new features which interact with old features**. Imagine you can claim ownership of a page but to do so you need to provide identification. However, a new feature came out which enables you to upgrade your page to get extra benefits, but the only information required is valid payment data. Upon providing that they add you as owner of the page and you've bypassed the identification step. You'll learn as you continue reading a big part of my methodology is spending days/weeks understanding how the website should work and what the developers were expecting the user to input/do and then coming up with ways to break & bypass this.

Another great example of a simple business logic bug is being able to sign up for an account with the email example**@target.com**. Sometimes these accounts have special privileges such as no rate limiting and bypassing certain verifications.

Business/application logic vulnerabilities tend to surface after you've got an understanding of how the web application works and you have a clearer picture of what they have to offer. The more you use their website the more you'll start to understand how things **SHOULD** work (*as they've intended it to*), but do they actually function as intended? Imagine you have just won a competition on a website and you can visit /prize/claim to claim your prize. Is this endpoint (or the claim process) available to those who **haven't** won? Look for explicit warnings to describe how features should work as well as API docs and start poking!

Business/Application logic vulnerabilities are often overlooked as most people are spraying payloads hoping for the best, but when it comes to business/application logic issues typically there is no clear cut payload to use. It's actually **less** about the