My eyes tend to light up when faced against filters though. A filter means the parameter we are testing is vulnerable to XSS, but the developer has created a filter to prevent any malicious HTML. This is one of the main reasons I also spend a lot of time looking for XSS when first starting on a new program because if they are filtering certain payloads, **it can give you a feel for the overall security of their site**. Remember, XSS is the easiest bug type to prevent against, so why are they creating a filter? And what else have they created filters around (think SSRF.. filtering just internal IP addresses? Perhaps they forgot about http://169.254.169.254/latest/meta-data - chances are, they did!).

Process for testing for XSS & filtering:

Step One: Testing different encoding and checking for any weird behaviour

Finding out what payloads are allowed on the parameter we are testing and how the website reflects/handles it. Can I input the most basic <h2>, <img>, <table> without any filtering and it's reflected as HTML? Are they filtering malicious HTML? If it's reflected as &lt; or %3C then I will test for double encoding %253C and %26lt; to see how it handles those types of encoding. Some interesting encodings to try can be found on https://d3adend.org/xss/ghettoBypass. This step is about finding out what's allowed and isn't & how they handle our payload. For example if <script> was reflected as &lt;script&gt;, but %26lt;script%26gt; was reflected as <script>, then I know I am onto a bypass and I can begin to understand how they are handling encodings (which will help me in later bugs maybe!). If not matter what you try you always see &lt;script&gt; or %3Cscript%3E then the parameter in question may not be vulnerable.