As well as scanning for robots.txt on each subdomain it's time to start scanning for files and directories. Depending on if any file extensions have been revealed I will typically scan for the most common endpoints such as **/admin**, **/server-status** and expand my word list depending on the success. You can find wordlists referenced at the start of this guide as well as the tools used (FFuF, CommonSpeak).

Primarily you are looking for sensitive files & directories exposed but as explained at the start of this guide, **creating a custom wordlist as you hunt can help you find more endpoints to test for.** This is an area a lot of researchers have also automated and all they simply need to do is input the domain to scan and it will not only scan for commonly found endpoints, but it will also continuously check for any changes. **I highly recommend you look into doing the same as you progress**, it will aid you in your research and help save time. Spend time learning how wordlists are built as custom wordlists are vital to your research when wanting to discover more.

Our first initial look was to get a feel for how things work, and I mentioned writing down notes. Writing down parameters found (**especially vulnerable parameters**) is an important step when hunting and can really help you with saving you time. This is one reason I created “InputScanner” so I could easily scrape each endpoint for any input name/id listed on the page, test them & note down for future reference. I then used Burp Intruder again to quickly test for common parameters found across each endpoint discovered & test them for multiple vulnerabilities such as XSS. This helped me identify lots of vulnerabilities across wide-scopes very quickly with minimal effort. I define the position on **/endpoint** and then simply add discovered parameters onto the request, and from there I can use Grep to quickly check the results for any interesting behaviour. **/endpoint**?param1=xss”&param2=xss”. Lots of endpoints, lots of common parameters = bugs! (*Don’t forget to test GET.POST! I have had cases where it wasn’t vulnerable in a GET request but it was in a POST. $_GET vs $_POST*)