**Exploiting file name cache rules:**

Certain files such as robots.txt, index.html, and favicon.ico are common files found on web servers. They're often cached due to their infrequent changes. Cache rules target these files by matching the exact file name string.

To identify whether there is a file name cache rule, send a GET request for a possible file and see if the response is cached.

**Detecting normalization discrepancies**

To test how the origin server normalizes the URL path, use the same method that you used for static directory cache rules. For more information, see [Detecting normalization by the origin server](https://portswigger.net/web-security/web-cache-deception#detecting-normalization-by-the-origin-server).

To test how the cache normalizes the URL path, send a request with a path traversal sequence and an arbitrary directory before the file name. For example, /aaa%2f%2e%2e%2findex.html:

* If the response is cached, this indicates that the cache normalizes the path to /index.html.
* If the response isn't cached, this indicates that the cache doesn't decode the slash and resolve the dot-segment, interpreting the path as /profile%2f%2e%2e%2findex.html.

**\*\*Essentially we use the same path traversal based payload as detailed in “Exploiting Static Dir Cache Rules” doc. This time instead of a resources/assets directory we use a specific file that is commonly cached like favicon.ico or robots.txt -> this will be somewhat accounted for in the web cache deception scanner, but it could be good to test manually too. The files could potentially be in some sort of directory we may need to do something like /resources/favicon.ico hence why manual testing is still important here.\*\***

**Exploiting normalization discrepancies:**

Because the response is only cached if the request matches the exact file name, you can only exploit a discrepancy where the cache server resolves encoded dot-segments, but the origin server doesn't. Use the same method as for static directory cache rules - simply replace the static directory prefix with the file name. \*\*/sensitive<delim>%2f%2e%2e%2frobots.txt WE WILL ALSO NEED A DELIMETER RIGHT AFTER THE SENSATIVE ENDPOINT!!!!!\*\*