1. Basic SSRF against the local server:
   * Exploited the vulnerability by changing the stock check API to the localhost address and then administrator page was displayed.
   * Then from the source code we got the endpoint we need to hit in order to delete the user so we deleted the user from the loopback address.
2. Basic SSRF against another back-end system:
   * In this scenario we just had to mount a brute force attack on the last octet of the IPv4 address with the port number given and the admin page also given using BurpSuite’s Intruder.
3. SSRF with blacklist-based input filter:
   * The application in this case blocked the SSRF requests so, now we bypassed it by using URL encoding <http://127.0.0.1/admin> to http%3A%2F%2F127.1%2F%2561dmin and when we sent request to this. It worked well and we got the endpoint to hit in order to delete the account.
4. SSRF with whitelist-based input filter:
   * Now in this case the application is first extracting the host name and then validating it against a whitelist.
   * So, we’ll try to change the API call to <http://username@stock.weliketoshop.net/> and observed that this URL is accepted.
   * Then in the end we will put <http://localhost:80%2523@stock.weliketoshop.net/admin/delete?username=carlos> into the API to delete Carlos’s account where **%2523** is **#.**
5. SSRF with filter bypass via open redirection vulnerability:
   * In this case the backend server supports the URL redirection to so we constructed an URL that forces the server to send redirection to the given components.
   * By appending this in the original URL &path=http://192.168.0.68/admin.
6. Blind SSRF with out-of-band detection:
   * In this case we had to just insert an URL into the Referer header to force the application to interact with any out-of-band application so that we can trigger SSRF attacks.