1. Excessive trust in client-side controls:
   * The application had a vulnerability in the add to cart option, upon inspecting the add to cart request the price of the product goes within so we changed the price to very less and we were able to place order.
2. High-level logic vulnerability:
   * In this application we found a vulnerability in which we added an item of 1000 bucks and then added 8 items of 100 bucks and used BurpSuite’s Interceptor to modify the request of adding 8 items and changed it to subtract 8 items.
3. Low-level logic flaw:
   * In this case the vulnerability was that if we kept adding the same product to the cart again and again using BurpSuite’s Intruder using Null Payloads then the value of integer is exceeded and the price of the items returns to 0.
4. Inconsistent handling of exceptional input:
   * Started the content discoverer from the engagement tools on target’s site map and after a while found a page with **/admin** and we got to know that in order to access admin page we need to logged in as some particular email’s domain name. So, after registering myself as a user by inputting a long string we saw that the email which is stored was truncated to 255 characters only.
   * So, now we appended our email while registering in such a way that the system truncates the email as [something@dontwannacry.com](mailto:something@dontwannacry.com) so it allows us to access the administrator page.
5. Inconsistent security controls:
   * The vulnerability in this case was that, we need a dontwannacry.com’s email address in order to login as an administrator and after creating our account with some other email and after logging in we saw that we had an option to change the email address so we changed that address to the required one and got access to the admin page.
6. Weak isolation on dual-use endpoint:
   * Exploited the vulnerability by changing the password of administrator’s account by using Burpsuite’s Interceptor by modifying the username parameter to administrator after logging in as wiener because we were able to change the password without providing the current password by removing that field from the HTTP request completely.
7. Password reset broken logic:
   * The vulnerability was present in the password reset logic as it doesn’t check the **temp-forgot-password-token** present in the request as well as the URLwhenthenew password is sent through HTTP request.
   * So, we deleted the value of the token from both places and changed the username to the target’s username and sent the request using BurpSuite’s Repeater.
8. Insufficient workflow validation:
   * In this case I first purchased something I can afford and saw that to confirm the order the application sends a HTTP request without any information about the order and redirects us to an order confirmation page.
   * So, we added the out of budget thing to our cart and simply repeated the request of confirmation using Burpsuite’s Repeater and got the order confirmed.
9. Authentication bypass via flawed state machine:
   * Now in this scenario the application sends a request in which the role is defined after logging in so we just used BurpSuite’s Interceptor to drop the request and logged in as an administrator.
10. Flawed enforcement of business rules:
    * The vulnerability was in redeeming coupon codes section as there were two coupons present but when we tried adding the same coupon two times in a row then we got an error but when we proceeded alternatively, we were able to apply same coupon many times.
11. Infinite money logic flaw:
    * In this case we had a vulnerability in which the user can but a $10 coupon with a coupon code.
    * So, we can keep buying gift cards many times and every time he buys gift card and redeems it we get some amount of money in surplus.