- in POST-request, change the price of the product

- GET-request to 2FA-page to generate the 2FA-code for the victim, brute-force 2FA-code with POST-requests

- skip the 2FA-step

- try submitting negative values

- create integer overflow

- By registration, try very long email string so that it would be cut off

- change current email to administrative email

- try not submitting an input parameter value (URL parameters, POST parameters, cookies,..)

- also try deleting the input parameter name in the request (e.g., omit the password by login, omit current password when changing password)

- add a jacket in the cart, and GET the confirmation (skip other steps)

- drop the step for selecting role (accept the response to set new session cookie)

- use 2 coupon codes alternatively to reduce the price

- buy gift card with discount price, then redeem it (automate this process)

1. "Project options" > "Sessions" > "Session handling rules" > "Add" > "Scope" > "URL Scope" > "Include all URLs" > "Details" > "Rule actions" > "Add" > "Run a macro" > "Select macro" > "Add"
2. Select the following sequence of requests:
3. POST /cart
4. POST /cart/coupon
5. POST /cart/checkout
6. GET /cart/order-confirmation?order-confirmed=true

POST /gift-card

> "OK"

1. Select GET /cart/order-confirmation?order-confirmed=true > "Configure item" > "Add" to create a custom parameter. Name the parameter gift-card and highlight the gift card code at the bottom of the response > "OK" twice
2. Select POST /gift-card > "Configure item" > "Parameter handling" , use the drop-down menus to specify that the gift-card parameter should be derived from the prior response (response 6) > "OK".
3. Click "Test macro". Look at the response to GET /cart/order-confirmation?order-confirmation=true and note the gift card code that was generated. Look at the POST /gift-card request. Make sure that the gift-card parameter matches and confirm that it received a 302 response. Keep clicking "OK" until you get back to the main Burp window.
4. 415 null payload attacks **using 1 thread**

- find a place where inputs are encrypted and decrypted (e.g., error notifications), use this as an oracle to construct an encrypted token (e.g., to log in as administrator) (url decode => base64 decode => remove the unnecessary bytes => encode again)