1. Unprotected admin functionality:
   * We directly went to explore the **/robots.txt** link where we found the link to administrator panel and straightaway, we accessed it.
2. Unprotected admin functionality with unpredictable URL:
   * The application had a link to the administrator panel hidden in the source code of the home page so we logged in directly as administrator.
3. User role controlled by request parameter:
   * In this application the role was sent in the HTTP request’s cookie as **admin=false** so we used BurpSuite’s Interceptor to modify the request to **admin=true.**
4. User role can be modified in user profile:
   * In this case we noticed that whenever we try to update email, the request that is sent contains a role id parameter in it so we tried modifying it as the role id of administrator.
5. URL-based access control can be circumvented:
   * In this scenario we had an application which only had admin panel restriction implemented in the front-end and there are no access controls for it on the backend.
   * So, we tried using **X-Original-URL:** header in the HTTP request which asks the server to process the request from this only except from any other URL.
   * Now we changed the parameter to **/admin** and we are granted the access to the administrator’s page.
6. Method-based access control can be circumvented:
   * We first logged in as an administrator then tried promoting someone from user to admin privileges then sent the request to BurpSuite’s Repeater.
   * Now convert the POST request to GET request and change the session ID of the user to the admin’s and also change the username to the victim’s username.
7. User ID controlled by request parameter:
   * Exploited the vulnerability by logging in as our own account and in the URL change the id parameter to the victim’s id and extract the API key and submitted.
8. User ID controlled by request parameter, with unpredictable user IDs:
   * Firstly, we found any post by the victim then we studied the HTTP request and the response as well and found an id parameter and after logging in as our own account we changed our id parameter to victim’s.
9. User ID controlled by request parameter with data leakage in redirect:
   * We logged in using our own credentials and then and sent the request to BurpSuite’s Repeater and changed the id parameter to the victim’s and then even though the server redirected us to the home page but in the response, it gave use the id of the victim.
10. User ID controlled by request parameter with password disclosure:
    * Now in this case we logged in with our own credentials and then changed the id parameter in the URL to the administrator’s and then I the response we got the administrator’s password leaked.
11. Insecure direct object references:
    * Exploited the vulnerability by chatting with a bot and while downloading its transcript we noticed that the text files are downloaded with the name as an incrementing number so we changed the name of the file in the URL to send the download request using BurpSuite’s Repeater and got the password for the account.
12. Multi-step process with no access control on one step:
    * Exploited the vulnerability using the BurpSuite’s Repeater to replay the request by logging in as the administrator first and promoting a non-target guy as an administrator and then stealing the session ID of the admin and copying it into the request with the victim’s password.
13. Referrer-based access control:
    * We logged in as an administrator first and then promoted a non-target user to administrator role and then sending that request to the BurpSuite’s Repeater and then log in as the victim in an incognito browser and then replay the request after exchanging the session cookie with the non-admin user’s.