Access control

1. Lab: Unprotected admin functionality

В robots.txt лежал эндпоинт administrator-panel, который не защищен.

2. Lab: Unprotected admin functionality with unpredictable URL

В исходном коде изначальной страницы валялся код, создающий эндпоинт админской панели, все еще не защищенной

3. Lab: User role controlled by request parameter

В куки есть параметр Admin=false. Меняя на true можно гонять под админом

4. Lab: User role can be modified in user profile

В json’е обновления имейла можно вставить изменение параметра roleid, сделав его 2

5. Lab: User ID controlled by request parameter

Тут я могу посмотреть по ручке my-account?id=carlos и быть счастливым

6. Lab: User ID controlled by request parameter, with unpredictable user IDs

В постах на изнаальной странице можно найти что постил carlos и оттуда получить его uuid

7. Lab: User ID controlled by request parameter with data leakage in redirect

При попытке изменения айди и просмотра другого аккаунта редиректит, но выдает все содержимое страницы другого аккаунта

8. Lab: User ID controlled by request parameter with password disclosure

По ручке my-account?id=administrator можно найти кнопку смены пароля, которая в пост запросе в открытом виде передает пароль

9. Lab: Insecure direct object references

Нашел, что по ручке, скачивающей историю чата можно вбить айди чата. Вбил первое айди и там нашел пароль.

10. Lab: URL-based access control can be circumvented

Нашел, что блок идет по изначальной ручке, а не по X-Original-Url, так и получил изначальный доступ к админке. чтобы удалить пользователя надо примерно так же двигаться, только важно параметр передать в реальный запрос.

11. Lab: Method-based access control can be circumvented

почему то ручка на апгрейд блочит по посту, но не блочит не админов по гету

12. Lab: Multi-step process with no access control on one step

Примерно то же, что и в прошлый раз. только даже метод не надо менять. просто показывает, что если знаешь ручки, то можно чет поделать

13. Lab: Referer-based access control

Тут по реферреру можно проапгрейдить, причем ТОЛЬКО себя (ставим в реферрер админскую панель и все)

Server-side request forgery (SSRF)

1. Lab: Basic SSRF against the local server

Кнопка для получения “в наличии” имеет параметр, в который можно прокинуть ручку на удаление человека

2. Lab: Basic SSRF against another back-end system

Скриптом подобрал правильный адрес (192.168.0.32) и прогнал по прошлой схеме

3. Lab: Blind SSRF with out-of-band detection

В referer хедер вставил домен xxx.oastify.com и лаба решена

4. Lab: SSRF with blacklist-based input filter

узнал, что 127.1 то же, что и 127.0.0.1. И что если заюрлэнкодить a, то задача решается

5. Lab: SSRF with filter bypass via open redirection vulnerability

Параметр stockApi может редиректнуть, так что вставить туда параметр ниже и можно достичь желаемого.

```
stockApi=/product/nextProduct?path=http://192.168.0.12:8080/admin/delete?username=ca
```

6. Lab: Blind SSRF with Shellshock exploitation

Эксфильтрация через бурп коллаборатор

7. Lab: SSRF with whitelist-based input filter

Узнал, что # в url строке может откинуть остаток строки)

итого строкой ниже получилось удалить карлоса

```
stockApi=http://localhost:80%2523@stock.weliketoshop.net:8080/admin/delete?username=
```