do_you_know_proxy?

생성자	R Rena231
생성 일시	@2024년 5월 20일 오후 4:12
태그	

1. 문제

처음 웹페이지에 접근시 티켓 발급 및 티켓을 이용한 과자를 교환할 수 있는 홈페이지가 존재합니다.

또한 소개글에는

과자를 구매하기위해 만들어놓은 홈페이지입니다.
들리는 이야기에 따르면 관리자는 더많은 과자를 얻을수 있다고 합니다.
고객인 당신은 관리자인척 시스템을 속일 수 있을까요?

라고 적혀져 있는데 확인하면 고객(guest)인 당신은 관리자(admin)인척 시스템을 속일 수 있을까요? 정도로 해석이 가능하다.

또한 과자뽑기 버튼을 누르게 되면 티켓1개와 과자1개를 1:1로 교환해주는것을 확인할 수 있다. 소스코드를 살펴보면

```python
def issue_ticket(id):
    if 'tickets' not in session:
        session['tickets'] = 5
        session['snacks'] = 0

    if session['tickets'] > 0:
        with open('id.txt', 'r') as file:
            valid_ids = [line.strip() for line in file]

        if id == valid_ids[0]:
            session['tickets'] -= 1
            session['snacks'] += 1
            return f"티켓을 발급받았습니다. 현재 발급 가능 티켓 수: {session['tickets']}, 현재 과자 수: {session['snacks']}"
        elif id == valid_ids[1]:
            session['tickets'] -= 1
            session['snacks'] += 2
            return f"티켓을 발급받았습니다. 현재 발급 가능 티켓 수: {session['tickets']}, 현재 과자 수: {session['snacks']}"
        else:
            return "유효하지 않은 id입니다."
    else:
        return f"티켓이 모두 소진되었습니다. 현재 발급 가능 티켓 수: {session['tickets']}, 현재 과자 수: {session['snacks']}"
```

1:1 방식으로 교환해주는것과 예외로 1:2 형태로 티켓이 1개당 과자 2개를 교환해주는 코드가 존재하는데 문맥상 이 부분이 관리자를 위한 코드로 추측된다.

id.txt를 통해 vaild_ids에 리스트로 저장한후 그 값들을 대조하는 방식으로 보여지는데 막상 id.txt를 열어보면

id
id2

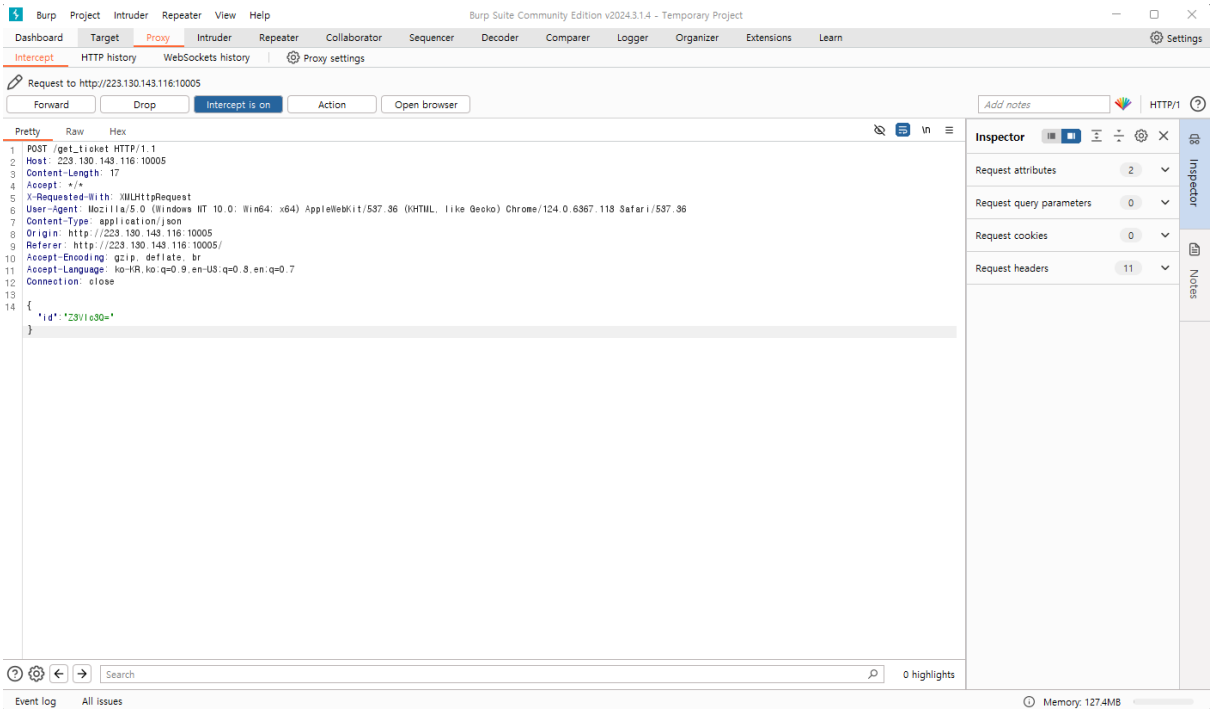
로 확인이 불가능하다.

2. 풀이

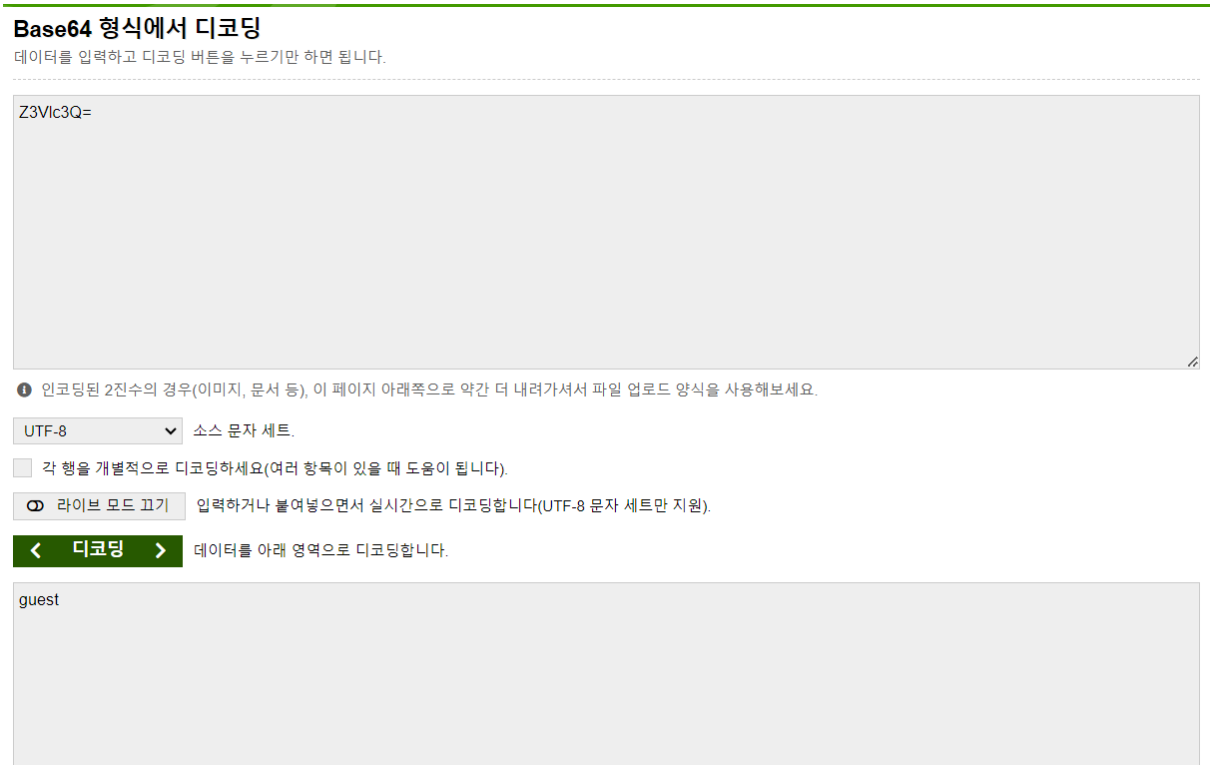
우선 코드를 보자 index.html을 살펴보게 되면 과자 뽑기 버튼을 클릭시 json 형태로 파라미터 id의 value인 guest를 base64 인코딩 한 후 전송하는 코드가 존재한다.

```
$(document).ready(function() {
    // 과자 뽑기 버튼 클릭 이벤트
    $('#get-ticket-btn').click(function() {
        var id = 'guest';
        var encodedId = btoa(id); // ID 값을 Base64로 인코딩
        var data = { id: encodedId }; // JSON 데이터 생성
        $.ajax({
            url: '/get_ticket',
            type: 'POST',
            contentType: 'application/json', // Content-Type을 JSON으로 설정
            data: JSON.stringify(data), // 데이터를 JSON 문자열로 변환하여 전송
            success: function(response) {
                alert(response.message);
                location.reload(); // 페이지 새로고침
            },
            error: function(xhr, status, error) {
                alert(xhr.responseJSON.message);
                location.reload(); // 페이지 새로고침
            }
        });
    });
```

그렇다면 티켓 발급 버튼을 누를때는 무슨일이 일어나는지 Burp 혹은 피들러로 확인해보자.



Json 형식으로 /get_ticket 엔드포인트에 어떠한 내용을 보내고 있는것 같다. 소스코드에서는 btoa(base64 Encoding)으로 전달하는 방식이라 했으니 Z3Vlc3Q=를 디코딩해보면 'guest'로 확인된다.



그렇다면 id.txt의 값중 id나 id2 둘중하나에는 Z3Vlc3Q=가 포함되어 있다는 의미인데 우리는 guest가 아닌 어떠한 다른 인자를 넣었을때 티켓1개당 과자2개를 받는 값을 알아내야 한다.

다행히도 그에대한 힌트는 문제에서 확인이 가능하다. 고객(guest)와 관리자(admin)으로 대략적으로 추측이 가능하다.

admin을 인코딩한다음 그값을 넣어서 확인해보자. 역시나 의도대로 실행된다. 이렇게되면 id.txt는 guest와 admin의 base64 인코딩값인 Z3Vlc3Q=와 YWRtaW4=인 것을 확인할 수 있다.

이를 이용해서 계속해서 admin으로 변조해 요청하면 과자의 개수를 5개 이상으로 만들 수 있다. 소스코드에서는 get_snacks_count가 10개 이상이면 flag를 제출한다는 코드의 내용을 확인할 수 있다.

```python
@app.route('/reset_ticket', methods=['POST'])
def reset_ticket_route():
    try:
        result = reset_tickets()
        return jsonify({"message": result, "ticket_count": get_ticket_count(), "snack_count": get_snacks_count()})
    except Exception as e:
        return jsonify({"message": str(e), "ticket_count": get_ticket_count(), "snack_count": get_snacks_count()}), 500


@app.route('/trade_snack', methods=['POST'])
def trade_snack():
    if get_snacks_count() >= 10:
        flag_content = read_flag()
        return jsonify({"message": "You have enough snacks!", "flag": flag_content})
    else:
        return jsonify({"message": "You need more snacks to get the flag."})

def read_flag():
    with open('flag.txt', 'r') as file:
        flag_content = file.read()
    return flag_content

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80, debug=False)
```

예상대로 id가 admin의 값으로 10개를 만든 후 과자 교환 버튼을 누르면 성공적으로 flag가 노출된다.