

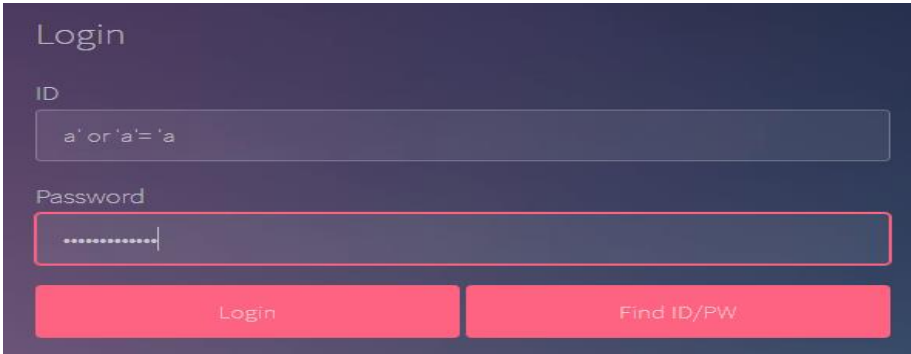
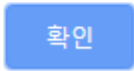
PosTrip

1. SQL Injection
2. 크로스 사이트 스크립팅(XSS)
3. 메모리 버퍼 오버플로우
4. 위험한 형식 파일업로드
5. 사용자 하드디스크에 저장되는 쿠키를 통한 정보노출
6. 주석문안에 포함된 시스템 주요 정보
7. 경로조작 및 자원삽입
8. 오류메시지를 통한 정보노출
9. 적절한 인증 없는 중요기능 허용
10. 반복된 인증시도 제한기능 부재
11. 약한 문자열 강도
12. CSRF

Web 취약점 분석 평가 항목

점검항목	항목중요도	항목코드
1. SQL Injection	상	
2. 크로스사이트 스크립팅(XSS)	중	
3. 위험한 형식 파일 업로드	상	
4. 메모리 버퍼 오버플로우	상	
5. 사용자 하드디스크에 저장되는 쿠키를 통한 정보 노출	상	
6. 주석문 안에 포함된 시스템 주요정보	중	
7. 경로 조작 및 자원 삽입	상	
8. 오류 메시지를 통한 정보 노출	상	
9. 적절한 인증 없는 중요기능 허용	상	
10. 반복된 인증시도 제한 기능 부재	상	
11. 약한 문자열 강도	상	
12. CSRF	상	

1. SQL Injection	
취약점 개요	
점검내용	<ul style="list-style-type: none"> 웹 페이지 내 SQL 인젝션 취약점 존재 여부 점검
점검목적	<ul style="list-style-type: none"> 대화형 웹 사이트에 비정상적인 사용자 입력 값 허용을 차단하여 악의적인 데이터베이스 접근 및 조작을 방지하기 위함
보안위험	<ul style="list-style-type: none"> 해당 취약점이 존재하는 경우 비정상적인 SQL 쿼리로 DBMS 및 데이터(Data)를 열람하거나 조작 가능하므로 사용자의 입력 값에 대한 필터링을 구현 하여야 함.
참고	※ SQL injection: 외부 입력 값을 쿼리 조작 문자열 포함 여부를 검증하지 않고 쿼리 작성 및 실행에 사용하는 경우, 쿼리의 구조와 의미가 변경 되서 실행되는 것 ※ SQL injection 공격 관련 코드 검토 필요 ※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> 소스코드, 웹 방화벽
판단기준	양호 : 임의의 SQL Query 입력에 대한 검증이 이루어지는 경우
	취약 : 임의의 SQL Query 입력에 대한 검증이 이루어지지 않은 경우

조치방법	<ul style="list-style-type: none"> ● 정적 쿼리를 사용, 구조화된 쿼리 실행, 파라미터화된 쿼리 실행, 입력값에 따라 쿼리문의 구조가 바뀌지 않도록 한다. ● ORM 프레임워크를 사용하는 경우, 외부 입력 값을 쿼리맵에 바인딩할 때 반드시 #기호를 이용한다. ● 입력값을 검증 → 외부 입력값에 쿼리 조작 문자열 포함 여부를 검증 후 쿼리문 생성 및 실행에 사용한다. ● 오류 메시지에 시스템 정보가 노출되지 않도록 한다. ⇒ Error-based SQL Injection 공격을 완화 ● DB 사용자의 권한을 최소로 부여한다. = 해당 어플리케이션에서 사용하는 DB 객체에 대해서만 권한을 부여한다. ⇒ Stored Procedure 또는 UNION-based SQL Injection 공격을 완화
점검 및 조치 사례	
<ul style="list-style-type: none"> ● 점검방법 <p>Step 1) 로그인 창에 참이 되는 SQL 쿼리를 전달하여 로그인 되는 확인</p>  <p>Step 2) 잘 막아져있다.</p> <p>"등록되지않은 사용자 이거나 비밀번호가 틀렸습니다."</p>  ● 보안대책방안 <p>☆ 입력 값 검증을 통해 쿼리문을 변경하는 것을 막는다.</p> 	

Login

✓ 로봇이 아닙니다. reCAPTCHA 개인정보 보호 - 약관

ID

'or'a'='a'

Password

비밀번호

Login Find ID/PW

localhost:3000 내용:

올바른 아이디를 입력해주세요.

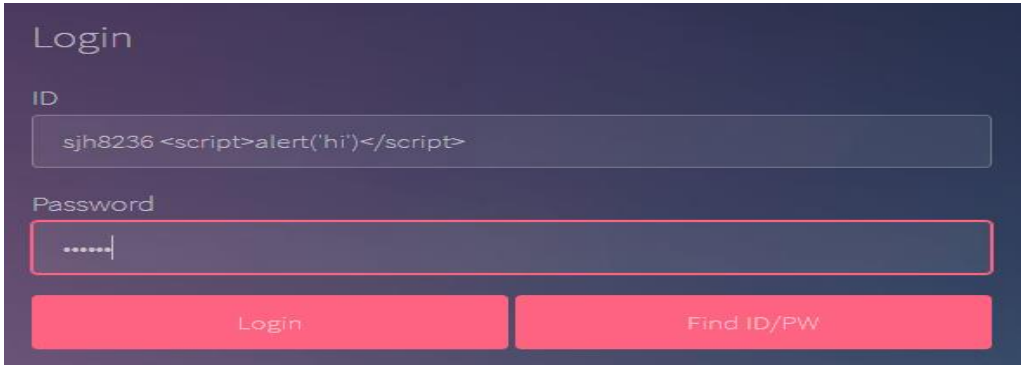
확인

```
let login_regece = /^[0-9a-zA-Z]*$/i;

if (login_id != "" && login_id.match(login_regece)) {
  $.post('/login_db', params, function (data, state) {
    alert(data);
    let DB = JSON.parse(data);
    location.replace('/mypage2');
  });
} else {
  if (checkbox=="pass"){
    alert('올바른 아이디를 입력해주세요.');
```

- ✧ 추가적으로 파라미터 값을 쿼리문에 바로 적용하지 않고 저장된 프로시저를 사용하여 처리하거나, ORM 프레임워크를 통해 쿼리문을 수행하도록 한다.

```
return User.findAll({
  attributes: ['db_count', 'db_time'],
  where: {
    db_id: req.body.login_id,
  }
}).then((result) => {
  let DB2 = JSON.stringify(result);
  let DB1 = JSON.parse(DB2);
  let DB = DB1[0];
  rs = DB.db_count;
  tm = DB.db_time;
  if (rs > 4) {
    time_check()
  } else {
    login_count();
    res.send('등록되지않은 사용자 이거나 비밀번호가 틀렸습니다.');
```

2. 크로스사이트 스크립팅(XSS)	
취약점 개요	
점검내용	<ul style="list-style-type: none"> 웹 페이지 내 크로스사이트 스크립팅 취약점 존재 여부 점검
점검목적	<ul style="list-style-type: none"> 웹 페이지 내 크로스사이트 스크립팅 취약점을 제거하여 악성 스크립트의 실행을 차단
보안위험	<ul style="list-style-type: none"> 웹 애플리케이션에서 사용자 입력 인수 값에 대한 필터링이 제대로 이루어지지 않을 경우, 사용자 인수 값을 받는 웹 사이트 게시판, URL 등에 악의적인 스크립트를 삽입하여 게시글이나 이메일을 읽는 사용자의 쿠키(세션)를 도용하거나 악성코드(URL 리다이렉트)를 유포할 수 있음.
참고	※ 크로스사이트 스크립팅: 악의적인 사용자가 공격하려는 사이트에 스크립트를 넣는 기법으로 공격 방식은 크게 Reflective xss, Stored xss, DOM xss 가 있다. ※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> 소스코드, 웹 방화벽
판단기준	양호 : 사용자 입력 인수 값에 대한 검증 및 필터링이 이루어지는 경우
	취약 : 사용자 입력 값에 대한 검증 및 필터링이 이루어지지 않으며, HTML코드가 입력 실행되는 경우
조치방법	<ul style="list-style-type: none"> 웹 사이트의 게시판, 자료실, URL 등에서 사용자로부터 입력받는 인수 값에 대해 검증 로직을 추가하거나 인수 값이 입력되더라도 실행되지 않게 하고, 부득이하게 게시판에서 HTML 을 사용하는 경우 HTML 코드 중 필요한 코드에 대해서만 입력 가능하도록 설정
점검 및 조치 사례	
<ul style="list-style-type: none"> 점검방법 <p>Step 1) id 칸에 <script>를 삽입한다.</p> 	

Step 2) 잘 막아져 있다.

"등록되지 않은 사용자이거나 비밀번호가 틀렸습니다."

확인

- 보안대책방안

- ✧ 입력 값 파라미터에 대한 정규식 검증을 하도록한다.

특수문자나 스크립트 언어, 인코딩된 문자 등에 대해 검증하여 검증되지 않은 입력 시 알림을 주도록 하였다.

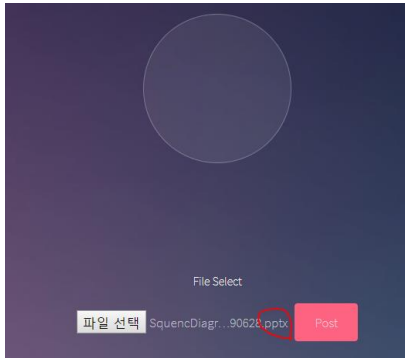
The screenshot shows a web application interface with a dark blue background. At the top, there is a navigation bar with links: HOME, Login, Join, MyPage, and FAQ. A white alert box in the top center displays the text: "localhost:3000 내용: 올바른 아이디를 입력해주세요." (Content of localhost:3000: Please enter a valid ID). Below the alert box, there is a "확인" (Confirm) button. The main content area is titled "Login". It features a reCAPTCHA challenge with a green checkmark and the text "로봇이 아닙니다." (I am not a robot). Below the reCAPTCHA, there are two input fields: "ID" and "Password". The "ID" field contains the text "'or 'a'='a'". The "Password" field contains the text "비밀번호" (Password). At the bottom of the login form, there are two buttons: "Login" and "Find ID/PW".

The screenshot shows a web application interface with a dark blue background. At the top, there is a navigation bar with links: HOME, Login, Join, MyPage, and FAQ. A white alert box in the top center displays the text: "localhost:3000 내용: 올바른 아이디를 입력해주세요." (Content of localhost:3000: Please enter a valid ID). Below the alert box, there is a "확인" (Confirm) button. The main content area is titled "Login". It features a reCAPTCHA challenge with a green checkmark and the text "로봇이 아닙니다." (I am not a robot). Below the reCAPTCHA, there are two input fields: "ID" and "Password". The "ID" field contains the text "haha <script>session.cookie</script>". The "Password" field contains the text "비밀번호" (Password). At the bottom of the login form, there are two buttons: "Login" and "Find ID/PW".

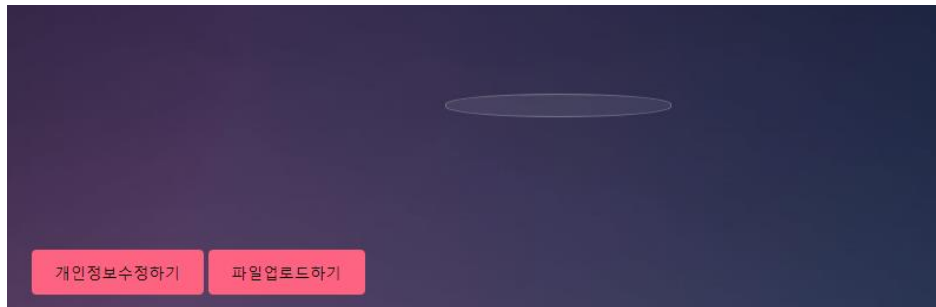
localhost:3000 내용:

올바른 아이디를 입력해주세요.

확인

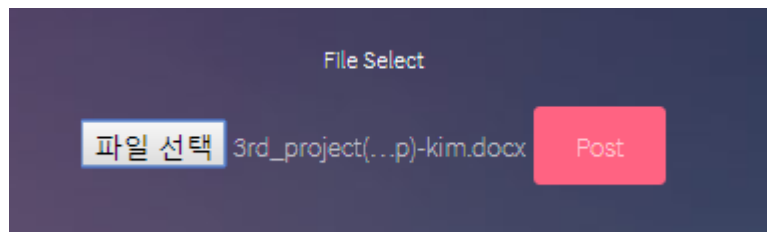
3. 위험한 형식 파일 업로드	
취약점 개요	
점검내용	<ul style="list-style-type: none"> 웹 사이트 게시판, 자료실에 부적절한 형식의 파일 업로드 및 실행 가능 여부 점검
점검목적	<ul style="list-style-type: none"> 업로드 되는 파일의 확장자에 대한 적절성 여부를 검증하는 로직을 통해 공격자가 조작된 Server Side Script 파일 업로드 방지 및 서버 상에 저장된 경로를 유추하여 해당 Server Side Script 파일 실행을 불가능하게 하기 위함.
보안위협	<ul style="list-style-type: none"> 해당 취약점 존재 시 공격자가 조작된 Server Side Script 파일을 업로드 하고 실행하여, 셸 권한 획득 후 홈페이지를 통해 시스템 명령어를 실행하고, 웹 브라우저를 통해 그 결과 값을 보며, 시스템 관리자 권한 획득 또는 인접 서버에 대한 침입을 시도할 수 있음.
참고	※ Server Side Script : 웹에서 사용되는 스크립트 언어 중 서버 사이드에서 실행되는 스크립트 ※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> 소스코드, Web Server
판단기준	양호 : 업로드 되는 파일에 대한 확장자 검증이 이루어지는 경우
	취약 : 업로드 되는 파일에 대한 확장자 검증이 이루어지지 않는 경우
조치방법	<ul style="list-style-type: none"> 업로드 되는 파일에 대한 확장자 검증 및 실행 권한 제거
점검 및 조치 사례	
<ul style="list-style-type: none"> 점검방법 <div>Step 1) 부적절한 사진 형식인 pptx 업로드</div> <div>  </div> 	

Step 2) 업로드 실행이 안 된 채로 표현



- 보안대책방안

- ✧ 서비스에 맞는 파일 형식 (Ex. 블로그 포스팅: Image(jpg, jpeg, gif 등))만 업로드 할 수 있도록 필터링 한다.



localhost:3000 내용:

이미지 파일만 업로드 가능합니다.

확인

4. 메모리 버퍼 오버플로우	
취약점 개요	
점검내용	<ul style="list-style-type: none"> ● 파라미터 입력 값에 대한 적절성 점검 여부 진단
점검목적	<ul style="list-style-type: none"> ● 애플리케이션에서 파라미터 입력 값에 대한 적절성을 점검하여 비정상적 오류 발생을 차단하기 위함
보안위험	<ul style="list-style-type: none"> ● 애플리케이션 입력 값의 크기에 대한 적절성이 검증되지 않을 경우 개발 시에 할당된 저장 공간보다 더 큰 값의 입력이 가능하고 이로 인한 오류 발생으로 의도되지 않은 정보 노출, 프로그램에 대한 비인가된 접근 및 사용 등이 발생할 수 있음
참고	※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> ● 소스코드
판단기준	양호 : 파라미터 입력 값에 대량의 인수 값 전달 시 에러 페이지나 오류가 발생되지 않는 경우
	취약 : 파라미터 입력 값에 대한 검증이 이루어지지 않고 에러 페이지나 오류가 발생하는 경우
조치방법	<ul style="list-style-type: none"> ● 외부 파라미터 입력 값을 할당하여 사용하는 경우 변수에 입력된 입력 값 범위를 검사하여 외부 파라미터 입력 값이 허용 범위를 벗어나는 경우 에러 페이지가 반환 되지 않도록 조치
점검 및 조치 사례	
<ul style="list-style-type: none"> ● 점검방법 <p>Step 1) id 입력값은 varchar(20)으로 그 이상의값을 입력해 보았다.</p> <div data-bbox="373 1447 1187 1870" data-label="Form"> </div> <ul style="list-style-type: none"> ● 확대 <div data-bbox="264 1915 1299 2016" data-label="Form"> </div>	

- 결과:
아무 결과 반환 없이 DB에 등록도 되지 않는다.

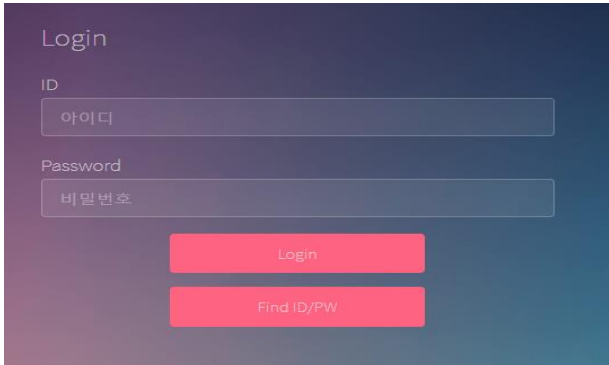
cf) 해당 정보 오류 메시지를 주었으면 좋겠음.

- 보안대책방안
 - ✧ 파라미터 값 검증을 통해서 일정 크기 이상의 값을 입력할 경우 알림을 주도록 한다.

localhost:3000 내용:

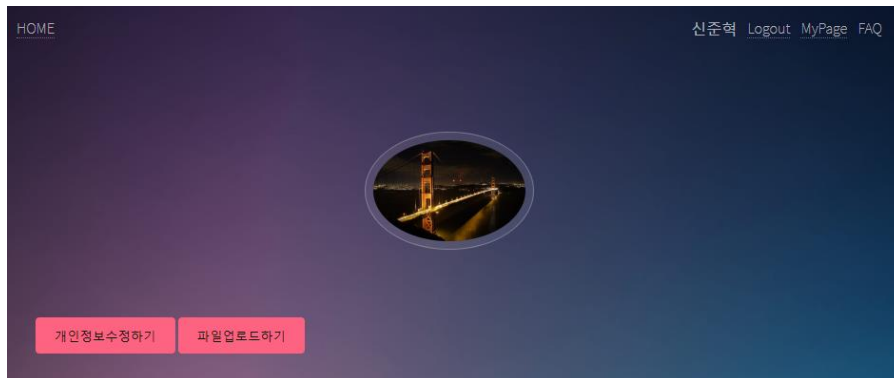
부적절한 회원정보를 입력하셨습니다. 다시 가입시도를 하시기 바랍니다.

확인

5. 사용자 하드디스크에 저장되는 쿠키를 통한 정보 노출(쿠키변조)	
취약점 개요	
점검내용	<ul style="list-style-type: none"> ● 쿠키 사용 여부 및 사용하는 경우 안전한 알고리즘으로 암호화 여부 점검
점검목적	<ul style="list-style-type: none"> ● 쿠키를 사용하는 경우 안전한 알고리즘으로 암호화하여 공격자가 쿠키 인젝션 등과 같은 쿠키 값 변조를 통한 다른 사용자로의 위장 및 권한 변경을 방지하고자 함.
보안위협	<ul style="list-style-type: none"> ● 쿠키(Cookie)는 클라이언트에 전달되는 값으로 중요 정보로 구성되므로 이 정보의 조작을 통해 다른 사용자의 유효한 세션을 취득할 수 있으며, 기타 중요 정보의 유출 및 변조가 발생할 위험이 존재
참고	※ 쿠키(Cookie): 인터넷 사용자가 어떠한 웹사이트를 방문할 경우 그 사이트가 사용하고 있는 서버에서 인터넷 사용자의 컴퓨터에 설치하는 작은 기록 정보 파일 ※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> ● 소스코드
판단기준	양호 : 쿠키를 사용하지 않고 Server Side Session을 사용하고 있거나, 쿠키(또는 Session)를 사용하는 경우 안전한 알고리즘(SEED, 3DES, AES)이 적용되어 있는 경우 취약 : 안전한 알고리즘이 적용되어 있지 않는 쿠키(Session)를 사용하거나, Client Side Session을 사용하는 경우
조치방법	<ul style="list-style-type: none"> ● 쿠키 대신 Server Side Session 방식을 사용하거나, 쿠키를 통해 인증 등 중요한 기능을 구현해야 할 경우엔 안전한 알고리즘(SEED, 3DES, AES) 적용
점검 및 조치 사례	
<ul style="list-style-type: none"> ● 점검방법 <p>Step 1) 처음 쿠키 값:</p> <p>s%3AP8GotvFvFhigzWitcO-7T5zAJtG8dYQH.aes4EsCeYLJhYjTRiVmD1XldQ6qAM%2ByCySWQkn9T4ng</p> 	

Step 2) 탈취한 쿠키 값:

s%3As_w7p5ELsCcJpRzkr0RZUJNaClc_YsHc.umw2DYhn6XoND5fYjjFT52DDZwvj5OVG8zPYaaG6jQc



Step 3) 탈취한 쿠키 값을 통해 회원정보에 접속할 수 있고, 중요정보 수정 및 삭제가 가능해진다.

- 보안대책방안

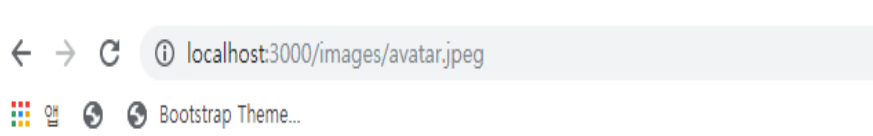
☆

6. 주석문 안에 포함된 시스템 주요정보	
취약점 개요	
점검내용	<ul style="list-style-type: none"> 주석문을 통해 확인할 수 있는 주요 정보 노출
점검목적	<ul style="list-style-type: none"> 주석문 안에 시스템 주요정보가 포함되어 있는 경우 공격자가 소스코드에 접근할 수 있다면 아주 쉽게 주요정보가 노출 될 수 있는 보안 취약점
보안위험	<ul style="list-style-type: none"> 공격자가 소스코드에 접근 할 수 있다면, 아주 쉽게 시스템에 침입할 수 있다.
참고	※ 주석문 안에 주요정보를 애초에 기입하지 않거나, 만약 기입했다면 반드시 제거해 주어야 한다.
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> 소스코드
판단기준	양호 : 주석문에 주요정보 기입한 것이 없음
	취약 : 주석문에 취약정보를 노출한 경우
조치방법	<ul style="list-style-type: none"> 주석문에 주요 정보를 제거한다.
점검 및 조치 사례	
<ul style="list-style-type: none"> 점검방법 <ul style="list-style-type: none"> 주석에 테이블 컬럼 명, 기타 다른 파라미터 이름을 노출할 수 있다. <pre> 1 const express = require('express'); 2 3 /* models 모듈에서 User 모델을 호출하여 User 로 선언 */ 4 var User = require('../models').User; 5 6 const router = express.Router(); 7 8 /* post 방식으로 호출된 router를 처리 */ 9 router.post('/', function (req, res, next) { 10 11 if (!req.body.login_id) { 12 res.json(JSON.stringify("아이디를 입력하세요")); 13 } else if (!req.body.login_pw) { 14 res.json(JSON.stringify("비밀번호를 입력하세요")); 15 } else { 16 /* User 테이블의 데이터를 가져오는 SQL문 */ 17 User.findAll({ 18 19 /* db_email, db_id, db_name, db_pw, db_birth, db_image 값을 가져옴 */ 20 attributes: ['db_email', 'db_id', 'db_name', 'db_pw', 'db_birth', 'db_image'], 21 22 /* 조건과 값이 일치하는 경우 */ 23 where: { 24 db_id: req.body.login_id, 25 db_pw: req.body.login_pw 26 } 27 }); </pre> <ul style="list-style-type: none"> 보안대처방안 <ul style="list-style-type: none"> 주석에는 필요한 내용만 넣거나, 파라미터 정보가 포함된 주석은 삭제한다. 	

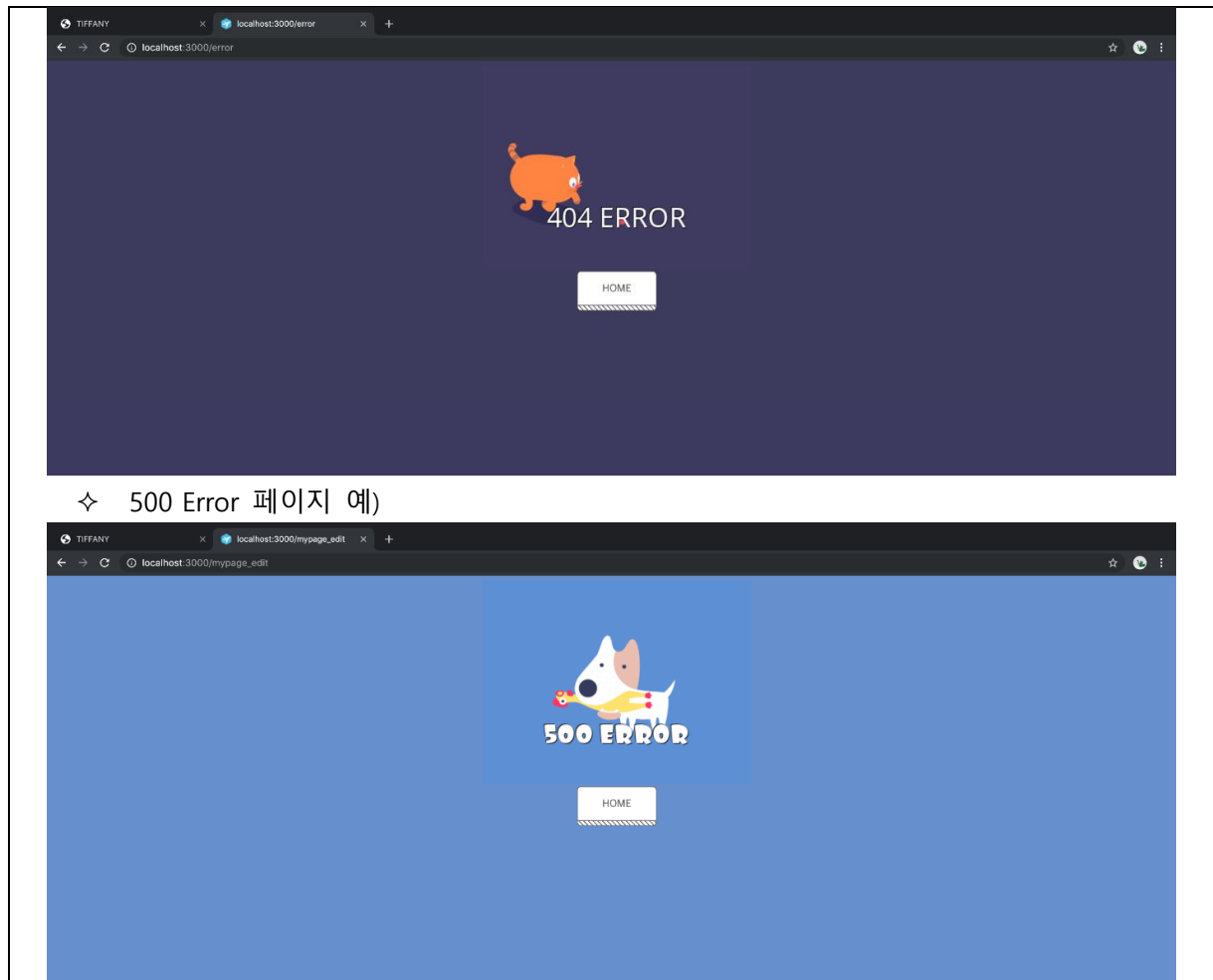
```
function login() {
  User.findAll({
    attributes: ['db_id', 'db_image', 'db_name', 'db_count', 'updatedAt'],
    /* 조건과 값이 일치하는 경우 */
    where: {
      db_id: req.body.login_id,
      db_pw: SHA256(req.body.login_pw)
    }
  })
  /* 조회 성공시 */
  .then((result) => {
    console.log("result : " + JSON.stringify(result));

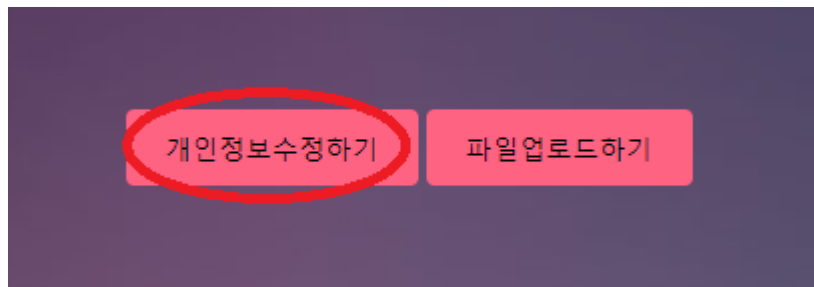
    let DB2 = JSON.stringify(result);
    let DB1 = JSON.parse(DB2);
    let DB = DB1[0];
    if (DB) {
      req.session.login = true;

      /* DB의 키 값이 존재하지 않을때 까지 반복 */
      for (key in DB) {
        count_reset();
      }
    }
  })
}
```

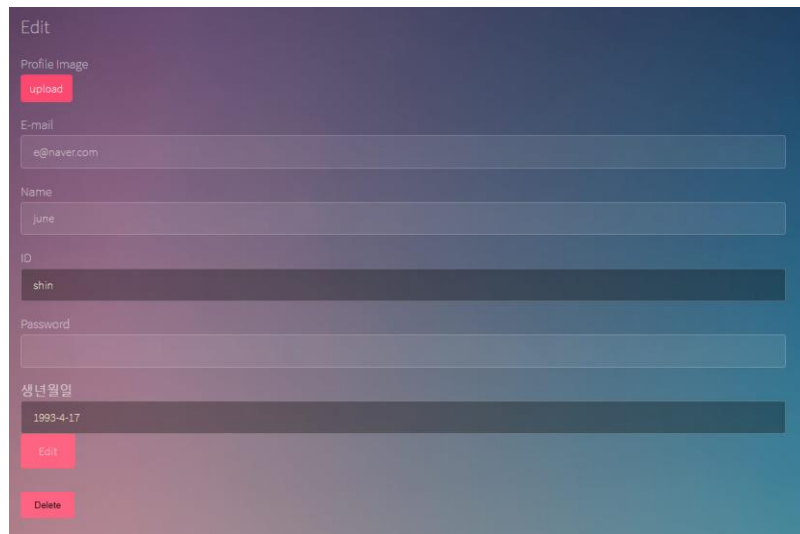

7. 경로 조작 및 자원삽입	
취약점 개요	
점검내용	<ul style="list-style-type: none"> ● 웹 서버와 웹 애플리케이션의 파일 또는 디렉터리의 접근 통제 여부 점검
점검목적	<ul style="list-style-type: none"> ● 웹 서버 또는 웹 애플리케이션의 중요한 파일과 데이터의 접근 및 실행을 방지 하고자 함.
보안위협	<ul style="list-style-type: none"> ● 웹 서버와 웹 애플리케이션의 파일 또는 디렉터리 접근이 통제되지 않아 웹 서버 또는 웹 애플리케이션의 중요한 파일과 데이터의 접근을 허용하는 취약점으로 웹 루트 디렉터리에서 외부의 파일까지 접근하여 이를 실행할 수 있음
참고	※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> ● 소스코드, Web Server, 웹 방화벽
판단기준	양호 : 웹사이트 루트 디렉터리 상위 디렉터리(예: /root) 접근이 불가능한 경우
	취약 : 웹사이트 루트 디렉터리 상위 디렉터리로 접근이 가능한 경우
조치방법	<ul style="list-style-type: none"> ● 웹 사이트의 최상위 디렉터를 웹 사이트 Root 디렉터리로 제한하여 웹사이트를 통해 웹 서버의 시스템 루트 디렉터리로 접근 못하게 제한
점검 및 조치 사례	
<ul style="list-style-type: none"> ● 점검방법 <div style="text-align: center;"> http://localhost:3000/../../../../images/avatar.jpeg http://localhost:3000/../../../../images/avatar.jpeg </div>  ● 보안대처방안 <ul style="list-style-type: none"> ✧ URL을 통해서 디렉토리 접근하는 것을 통제한다. ✧ ... 	

8. 오류 메시지를 통한 정보 노출	
취약점 개요	
점검내용	<ul style="list-style-type: none"> 에러 처리를 충분히 하지 않았을 때, 에러 정보에 중요정보가 포함되어 공격에 필요한 정보가 노출 될 수 있는 보안 취약점
점검목적	<ul style="list-style-type: none"> 예상치 못한 에러에 대한 로직 설계
보안위험	<ul style="list-style-type: none"> 충분치 않은 에러 메시지에 시스템의 내부정보 등 공격에 필요한 중요정보가 그대로 노출 될 수 있는 문제가 생길 수 있다.
참고	<p>※ 에러가 발생했을 때, 사용자에게 민감한 정보가 노출되지 않도록 미리 정의 된 메시지를 제공하는 에러처리 로직을 설계해야 한다. 이 때 오류메시지에는 정해진 사용자에게만 유용하도록 최소한의 정보만 포함해야 하는데 오류메시지에 개인정보, 시스템정보, 민감정보 등의 중요정보가 포함되지 않도록 시큐어코딩 규칙을 정의해야한다.</p>
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> 소스코드
판단기준	양호 : 중요 정보가 노출 되지 않는 페이지로 설계
	취약 : 중요 정보가 그대로 노출됨
조치방법	<ul style="list-style-type: none"> 로직 에러시 중요정보가 노출되지 않는 페이지를 만든다.
점검 및 조치 사례	
<ul style="list-style-type: none"> 점검방법 <p>Step 1) URL 주소 마지막에 /aaa 인 오류메시지를 입력해 정보를 노출시킨다.</p> <p>Not Found</p> <p>404</p> <pre> NotFoundError: Not Found at C:\lens\3_Postrip_김도원_김신\app.js:63:8 at Layer.handle [as handle_request] (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\layer.js:95:5) at trim_prefix (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:317:13) at C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:284:7 at Function.process_params (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:335:12) at next (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:275:10) at C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:635:15 at next (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:250:14) at Function.handle (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:174:3) at router (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:47:12) at Layer.handle [as handle_request] (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\layer.js:95:5) at trim_prefix (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:317:13) at C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:284:7 at Function.process_params (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:335:12) at next (C:\lens\3_Postrip_김도원_김신\node_modules\express\lib\router\index.js:275:10) at SendStream.error (C:\lens\3_Postrip_김도원_김신\node_modules\serve-static\index.js:121:7) at SendStream.emit (events.js:189:13) at SendStream.error (C:\lens\3_Postrip_김도원_김신\node_modules\send\index.js:270:17) at SendStream.onStatError (C:\lens\3_Postrip_김도원_김신\node_modules\send\index.js:421:12) at next (C:\lens\3_Postrip_김도원_김신\node_modules\send\index.js:738:16) at onstat (C:\lens\3_Postrip_김도원_김신\node_modules\send\index.js:725:14) at FSReqWrap.oncomplete (fs.js:153:21) </pre> 보안대처방안 <ul style="list-style-type: none"> ☆ 찾을 수 없는 페이지(404 Error), 서버 측 에러(500 Error)에 대한 표출 화면을 수정하여 제공한다. ☆ 404 Error 페이지 예) 	



9. 적절한 인증 없는 중요기능 허용	
취약점 개요	
점검내용	<ul style="list-style-type: none"> 회원정보 수정 시 적절한 인증이 부재함
점검목적	<ul style="list-style-type: none"> 개인정보 관리, 수정 시 점검을 요한다.
보안위협	<ul style="list-style-type: none"> 로그인 후 사용자 부재 시 타인이 회원정보를 변경 및 삭제 할 위험이 있다.
참고	※ 정보 수정 시 회원정보(아이디, 비밀번호)를 다시 한번 입력하게 한다.
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> 소스코드
판단기준	양호 : 정보수정 비밀번호를 요구한다.
	취약 : 아무런 제재 없이 회원 정보 수정이 된다.
조치방법	<ul style="list-style-type: none"> 정보수정 및 삭제 버튼 클릭 시 개인정보를 입력하도록 요구한다.
점검 및 조치 사례	
<ul style="list-style-type: none"> 점검방법 <p>Step 1) My page -> 개인정보수정하기</p> <div data-bbox="370 1424 1187 1709">  </div>	

Step 2) 추가적인 요청 없이 정보를 수정할 수 있게 된다.

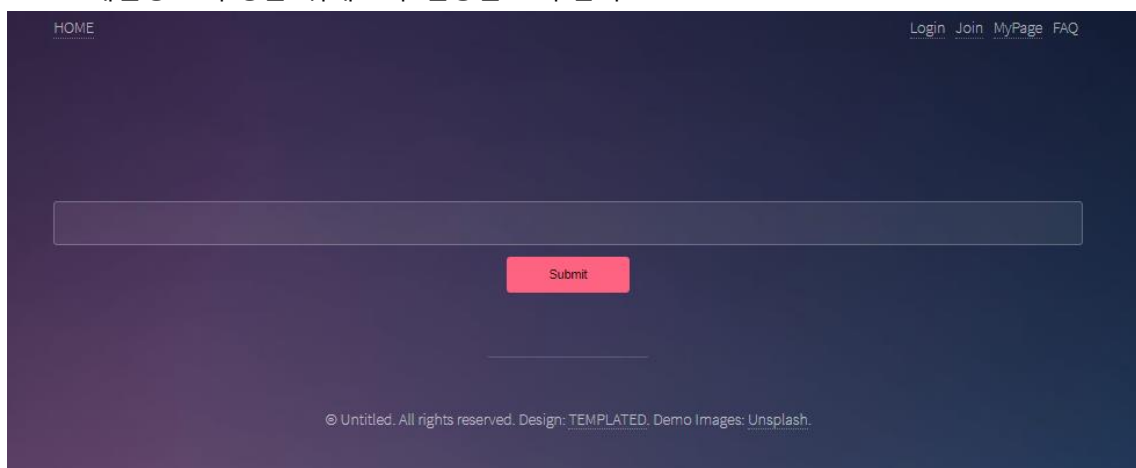


The screenshot shows a web form titled "Edit" for editing a user profile. It includes the following fields and controls:

- Profile Image:** A text input field with a red "upload" button next to it.
- E-mail:** A text input field containing the value "e@naver.com".
- Name:** A text input field containing the value "june".
- ID:** A text input field containing the value "shin".
- Password:** A text input field.
- 생년월일 (Date of Birth):** A text input field containing the value "1993-4-17".
- Buttons:** At the bottom of the form, there are two red buttons: "Edit" and "Delete".

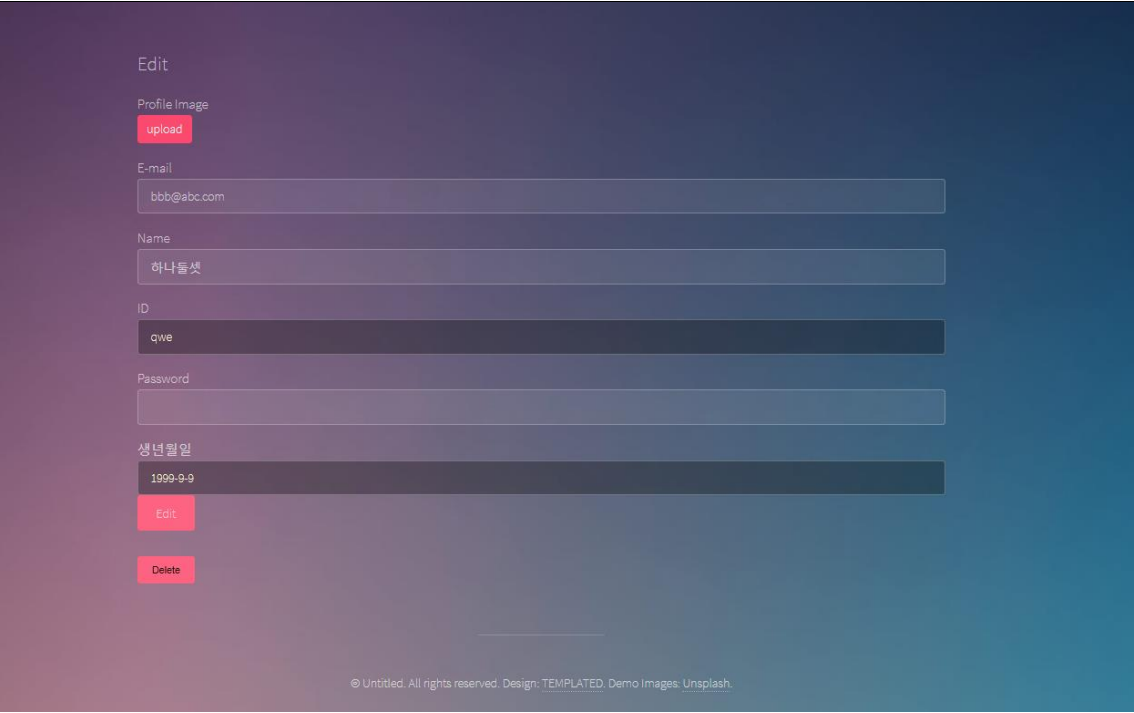
- 보안대책방안

- ✧ 개인정보 수정을 위해 2차 인증을 요구한다.



The screenshot shows a login page with a dark blue background. It features a single text input field for a username or password, and a red "Submit" button below it. The page includes a navigation bar at the top with links for "HOME", "Login", "Join", "MyPage", and "FAQ". At the bottom, there is a copyright notice: "© Untitled. All rights reserved. Design: TEMPLATED. Demo Images: Unsplash."

- ✧ 2차 인증 후 개인정보 수정페이지로 접근할 수 있게 하며, URL을 통해서 & 로그인을 하지 않고는 접속할 수 없도록 한다.



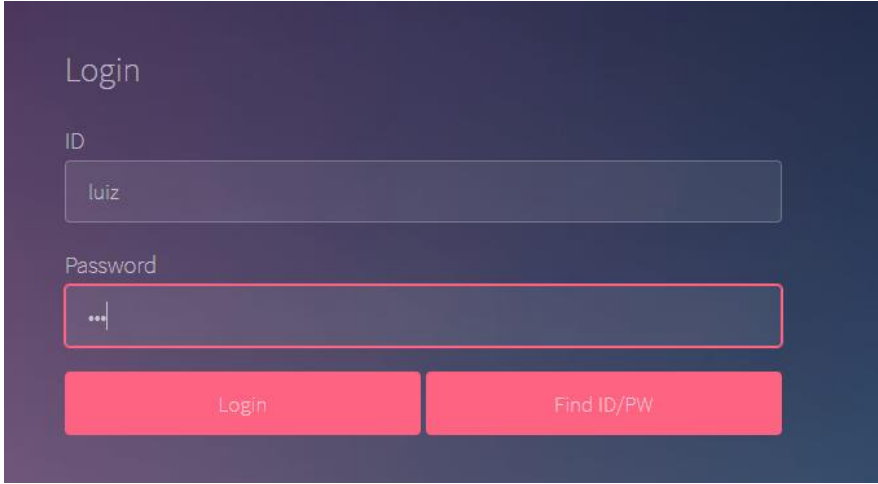
The screenshot shows a user profile editing interface. It includes fields for Profile Image (with an upload button), E-mail (bbb@abc.com), Name (하나둘셋), ID (qwe), Password, and Birthdate (1999-9-9). There are 'Edit' and 'Delete' buttons at the bottom. A copyright notice at the bottom reads: © Untitled. All rights reserved. Design: TEMPLATED. Demo Images: Unsplash.

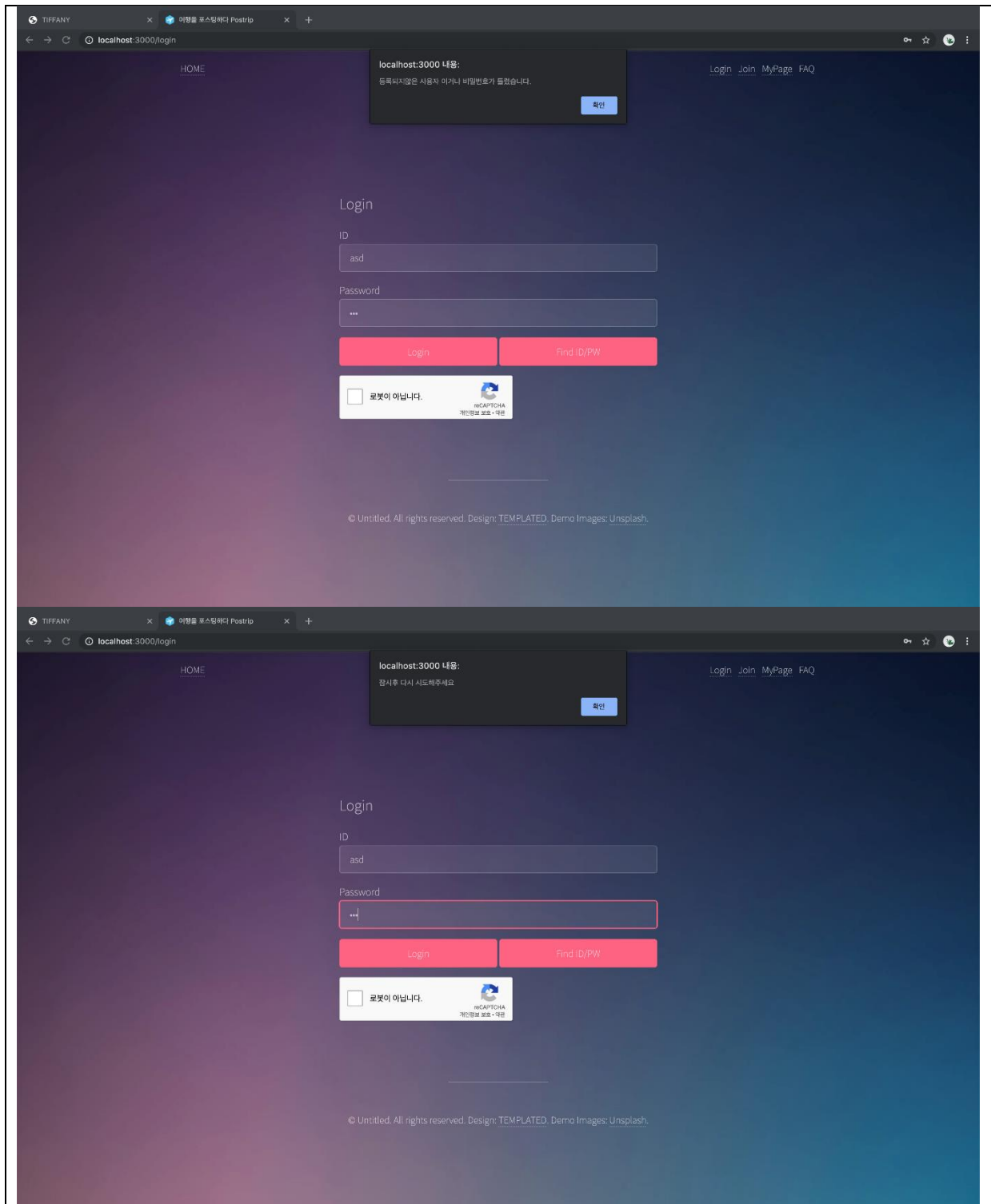
✧ 개인정보 수정에도 추가적인 인증(EX. 비밀번호 입력 등)을 통해서 수정할 수 있도록 한다.

localhost:3000 내용:

비밀번호를 입력해주세요.

확인

10. 반복된 인증시도 제한 기능 부재	
취약점 개요	
점검내용	<ul style="list-style-type: none"> 로그인 시 잘못된 정보를 계속 입력할 수 있다.
점검목적	<ul style="list-style-type: none"> 잘못된 정보로 로그인을 무한정 시도 할 수 있는 것을 방지한다.
보안위협	<ul style="list-style-type: none"> 시도 제한을 두지 않아 우연히 로그인에 성공되어 아이디와 비밀번호가 노출 될 가능성이 있다.
참고	※ 로그인 시도 제한을 경고하고 제한횟수가 몇 번 남았는지 팝업창으로 경고해준다.
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> 소스코드
판단기준	양호 : 제한횟수를 경고해주고, 횟수 초과시 경고 메시지를 준다.
	취약 : 제한횟수 없이 무한정 로그인이 가능하다.
조치방법	<ul style="list-style-type: none"> 로그인 기능에 횟수 제한을 걸어둔다. 처음부터 로그인 제한에 대한 경고 창을 준다. 제한횟수 초과시 서버를 다운시키거나, 공인증서 및 id/pw 찾기 페이지로 이동 시킨다.
점검 및 조치 사례	
<ul style="list-style-type: none"> 점검방법 <p>Step 1) 로그인 무한정 가능</p>  보안대책방안 	



MySQL Workbench interface showing a query result for the query: `SELECT * FROM postrip.users;`. The result grid displays 4 rows of data. The columns are: id, db_email, db_id, db_name, db_pw, db_birch, db_image, and db_count.

id	db_email	db_id	db_name	db_pw	db_birch	db_image	db_count
1	asd@naver.com	asd	asd	qwezxc123!	1999-5-5	1565343078624-1_ju0OfKrvDRZInTSG...	4

The Action Output pane shows the following actions:

Time	Action	Response	Duration / Fetch Time
18:06:51	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00029 sec / 0.0000...
18:07:00	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00024 sec / 0.000...
18:08:09	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00044 sec / 0.000...
18:08:19	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00036 sec / 0.000...
18:10:26	DROP TABLE 'postrip'. 'mypages', 'postrip...	0 row(s) affected	0.010 sec
18:33:25	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00056 sec / 0.000...

MySQL Workbench interface showing a query result for the query: `SELECT * FROM postrip.users;`. The result grid displays 0 rows of data. The columns are: id, db_email, db_id, db_name, db_pw, db_birch, db_image, and db_count.

id	db_email	db_id	db_name	db_pw	db_birch	db_image	db_count
1	asd@naver.com	asd	asd	qwezxc123!	1999-5-5	1565343078624-1_ju0OfKrvDRZInTSG...	0

The Action Output pane shows the following actions:

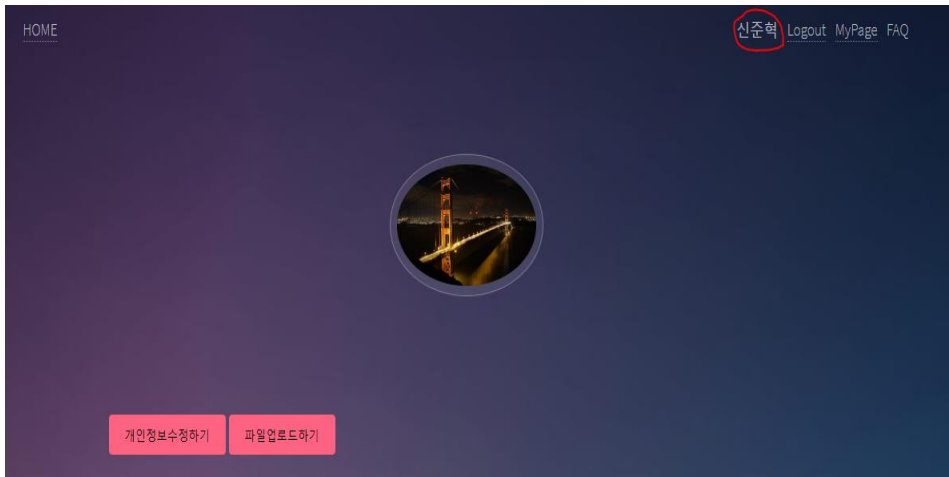
Time	Action	Response	Duration / Fetch Time
18:07:00	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00024 sec / 0.000...
18:08:09	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00044 sec / 0.000...
18:08:19	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00036 sec / 0.000...
18:10:26	DROP TABLE 'postrip'. 'mypages', 'postrip...	0 row(s) affected	0.010 sec
18:33:25	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00056 sec / 0.000...
18:35:02	SELECT * FROM postrip.users LIMIT 0, 1000	1 row(s) returned	0.00070 sec / 0.0000...

11. 약한 문자열 강도	
취약점 개요	
점검내용	<ul style="list-style-type: none"> 웹 페이지 내 로그인 폼 등에 약한 강도의 문자열 사용 여부 점검
점검목적	<ul style="list-style-type: none"> 유추 가능한 취약한 문자열 사용을 제한하여 계정 및 패스워드 추측 공격을 방지하기 위함
보안위협	<ul style="list-style-type: none"> 해당 취약점 존재 시 유추가 용이한 계정 및 패스워드의 사용으로 인한 사용자 권한 탈취 위험이 존재하며, 해당 위험을 방지하기 위해 값의 적절성 및 복잡성을 검증하는 체크 로직을 구현하여야 함.
참고	※ 약한 문자열 강도 취약점 : 웹 애플리케이션에서 회원가입 시 안전한 패스워드 규칙이 적용되지 않아 취약한 패스워드로 회원가입이 가능할 경우 공격자가 추측을 통한 대입 및 주변 정보를 수집하여 작성한 사전파일 통한 대입을 시도하여 사용자의 패스워드를 추출할 수 있는 취약점 ※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	<ul style="list-style-type: none"> 소스코드
판단기준	양호 : 관리자 계정(비밀번호 포함)이 유추하기 어려운 계정으로 설정되어 있는 경우
	취약: 관리자 계정(비밀번호 포함)이 유추하기 쉬운 계정으로 설정되어 있는 경우
조치방법	<ul style="list-style-type: none"> 계정 및 비밀번호의 체크 로직 추가 구현
점검 및 조치 사례	
<ul style="list-style-type: none"> 점검방법 <p>Step 1) 웹 사이트 로그인 페이지의 로그인 창에 추측 가능한 계정이나 패스워드를 입력하여 정상적으로 로그인 되는지 확인</p> <ul style="list-style-type: none"> 취약한 계정: admin, administrator, manager, guest, test, scott, tomcat, root, user 등 취약한 패스워드: abcd, aaaa, 1234, test, password, public 및 ID와 동일한 패스워드 - 	

The image shows a web login interface. At the top, it says 'Login'. Below that are two input fields. The first is labeled 'ID' and contains the text 'admin'. The second is labeled 'Password' and contains masked characters '****'. At the bottom of the form are two buttons: a red 'Login' button and a red 'Find ID/PW' button.

- 보안대처방안

Step 1) 취약한 계정 및 패스워드를 삭제하고, 사용자가 취약한 계정이나 패스워드 등록하지 못하도록 패스워드 규정이 반영된 체크 로직을 구현하도록 함

12. 크로스 사이트 리퀘스트 변조(CSRF)																																														
취약점 개요																																														
점검내용	● 사용자의 신뢰(인증) 정보의 변조 여부 점검																																													
점검목적	● 사용자 입력 값에 대한 적절한 필터링 및 인증에 대한 유효성을 검증하여 신뢰(인증) 정보 내의 요청(Request)에 대한 변조 방지																																													
보안위협	● 사용자의 신뢰(인증) 정보 내에서 사용자의 요청(Request)을 변조함으로써 해당 사용자의 권한으로 악의적인 공격을 수행할 수 있음																																													
참고	※ CSRF(Cross-site request forgery): 특정 사용자를 대상으로 하지 않고, 불특정 다수를 대상으로 로그인된 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록, 송금 등)를 하게 만드는 공격이다. ※ 소스코드 및 취약점 점검 필요																																													
점검대상 및 판단기준																																														
대상	● 소스코드, 웹 방화벽																																													
판단기준	양호 : 사용자 입력 값에 대한 검증 및 필터링이 이루어지는 경우																																													
	취약 : 사용자 입력 값에 대한 필터링이 이루어지지 않으며, HTML 코드(또는 스크립트)를 입력하여 실행되는 경우																																													
조치방법	● 사용자 입력 값에 대해 검증 로직 및 필터링 추가 적용																																													
점검 및 조치 사례																																														
● 점검방법																																														
Step 1) 웹 페이지에 로그인 해 접속을 한다.																																														
																																														
Step 1-2) DB에서 확인한다.																																														
<table><tr><th>id</th><th>db_email</th><th>db_id</th><th>db_name</th><th>db_pw</th><th>db_birth</th><th>db_image</th><th>createdAt</th><th>updatedAt</th></tr><tr><td>1</td><td>sjh8236@naver.com</td><td>sjh8236</td><td>신준혁</td><td>123123</td><td>1992-9-12</td><td>1565230831974-sanfran.jpg</td><td>2019-08-08 02:20:31</td><td>2019-08-08 02:20:31</td></tr><tr><td>3</td><td>k@naver.com</td><td>shin</td><td>아아</td><td>123123</td><td>1992-7-7</td><td>1565247808624-SauecDiagram신준혁190628.pptx</td><td>2019-08-08 07:03:23</td><td>2019-08-08 07:03:23</td></tr><tr><td>4</td><td>prague@naver.com</td><td>praha</td><td>cezech</td><td>123123</td><td>1944-7-26</td><td>1565250765055-paris.jpg</td><td>2019-08-08 07:52:45</td><td>2019-08-08 07:52:45</td></tr><tr><td>5</td><td>admin@postrip.com</td><td>admin</td><td>administrator</td><td>aaaa</td><td>1592-10-21</td><td>1565253361162-꽃.jpg</td><td>2019-08-08 08:36:01</td><td>2019-08-08 08:36:01</td></tr></table>		id	db_email	db_id	db_name	db_pw	db_birth	db_image	createdAt	updatedAt	1	sjh8236@naver.com	sjh8236	신준혁	123123	1992-9-12	1565230831974-sanfran.jpg	2019-08-08 02:20:31	2019-08-08 02:20:31	3	k@naver.com	shin	아아	123123	1992-7-7	1565247808624-SauecDiagram신준혁190628.pptx	2019-08-08 07:03:23	2019-08-08 07:03:23	4	prague@naver.com	praha	cezech	123123	1944-7-26	1565250765055-paris.jpg	2019-08-08 07:52:45	2019-08-08 07:52:45	5	admin@postrip.com	admin	administrator	aaaa	1592-10-21	1565253361162-꽃.jpg	2019-08-08 08:36:01	2019-08-08 08:36:01
id	db_email	db_id	db_name	db_pw	db_birth	db_image	createdAt	updatedAt																																						
1	sjh8236@naver.com	sjh8236	신준혁	123123	1992-9-12	1565230831974-sanfran.jpg	2019-08-08 02:20:31	2019-08-08 02:20:31																																						
3	k@naver.com	shin	아아	123123	1992-7-7	1565247808624-SauecDiagram신준혁190628.pptx	2019-08-08 07:03:23	2019-08-08 07:03:23																																						
4	prague@naver.com	praha	cezech	123123	1944-7-26	1565250765055-paris.jpg	2019-08-08 07:52:45	2019-08-08 07:52:45																																						
5	admin@postrip.com	admin	administrator	aaaa	1592-10-21	1565253361162-꽃.jpg	2019-08-08 08:36:01	2019-08-08 08:36:01																																						

Step 2) 크로스 사이트 스크립팅이 취약한 페이지에서 CSRF 할 게시글을 등록한다.

제목	CSRF-postrip
내용	<pre><form action="http://70.12.227.192:3000/delete_db" method="get" enctype="multipart/form-data"> <input id="delete_btn" type="submit" value="Delete" /> </form> <script>document.getElementById("delete_btn").click();</script></pre>
파일	<input type="text"/> <input type="button" value="찾아보기..."/> <small>* 임의로 파일명이 변경될 수 있습니다.</small>

Step 3) 작성한 게시글을 클릭하도록 한다.

1	CSRF-postrip	관리자	0	0	0	2019-08-09 00:00:00
---	--------------	-----	---	---	---	------------------------

Step 4) '삭제 완료' 문구가 보인다.

"{msg: '삭제 완료'}"

Step 5) DB에서도 사라진 것을 볼 수 있다.

id	db_email	db_id	db_name	db_pw	db_birth	db_image	createdAt	updatedAt
3	k@naver.com	shin	아아	123123	1992-7-7	1565247809624-SequenceDiagram신준혁190628.pptx	2019-08-08 07:03:23	2019-08-08 07:03:23
4	prague@naver.com	praha	cezh	123123	1944-7-26	1565250765055-paris.jpg	2019-08-08 07:52:45	2019-08-08 07:52:45
5	admin@postrip.com	admin	administrator	aaaa	1592-10-21	1565253361162-꽃.jpg	2019-08-08 08:36:01	2019-08-08 08:36:01

● 보안대책방안

Step 1)

node 에서는 내부 설치를 통해 CSRF를 막을 수 있다.

(참조 : <https://www.npmjs.com/package/csrf>)

Installation

This is a **Node.js** module available through the **npm registry**. Installation is done using the **npm install command**:

```
$ npm install csrf
```

Step 2) login.js 에 csrf 코드를 삽입해준다.

```
var csrf = require('csrf');

var csrfProtection = csrf();
router.use(csrfProtection);

router.get('/login', function(req,res,next) {
  res.render('/login', {csrfToken: req.csrfToken()});
})
```

Step 3) login.ejs 에 csrfToken 을 부여한다.

```
<input type="hidden" name="_csrf" value="{{ csrfToken }}">
```

