SINGAPORE POLYTECHNIC SP

DIPLOMA IN INFORMATION TECHNOLOGY

Enterprise System Development (ST0505)

CA1 Assignment - Report
Cover Page

Done By: Gao QianXi
Class: DIT/FT/2A/04

Table of Contents

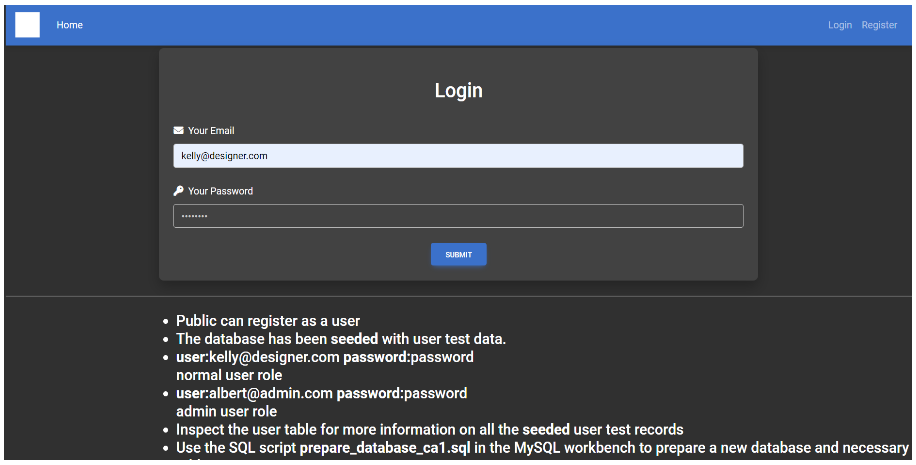
1. Cross-site Scripting (XSS)

1.1 Definition

It occurs when untrusted data is included in a new web page without proper validation or escaping, allowing attackers to execute malicious scripts in the victim's browser.

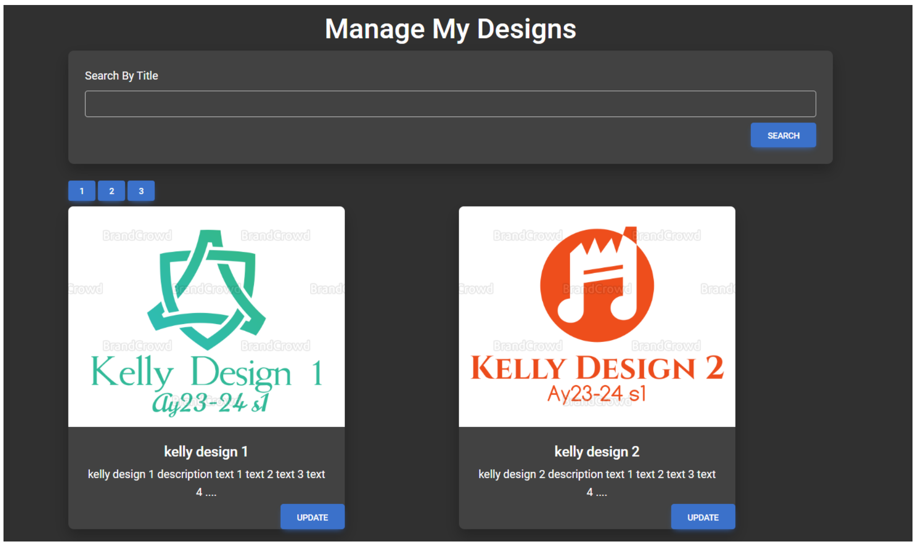
1.2 How to exploit vulnerability - Reflected XSS

Step 1: Login as kelly



- Public can register as a user
- The database has been **seeded** with user test data.
- **user:**kelly@designer.com **password:**password
  normal user role
- **user:**albert@admin.com **password:**password
  admin user role
- Inspect the user table for more information on all the **seeded** user test records
- Use the SQL script **prepare_database_ca1.sql** in the MySQL workbench to prepare a new database and necessary

Step 2: Click on "search" -> On kelly design 1 click "update"



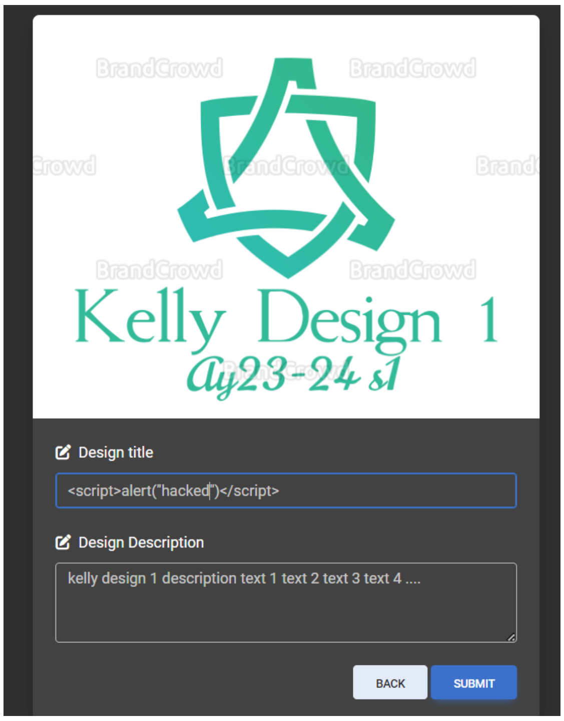
kelly design 1

kelly design 1 description text 1 text 2 text 3 text 4 ....

kelly design 2

kelly design 2 description text 1 text 2 text 3 text 4 ....

Step 3: In the design title text box type “<script>alert("hacked")</script>” -> Click “submit”



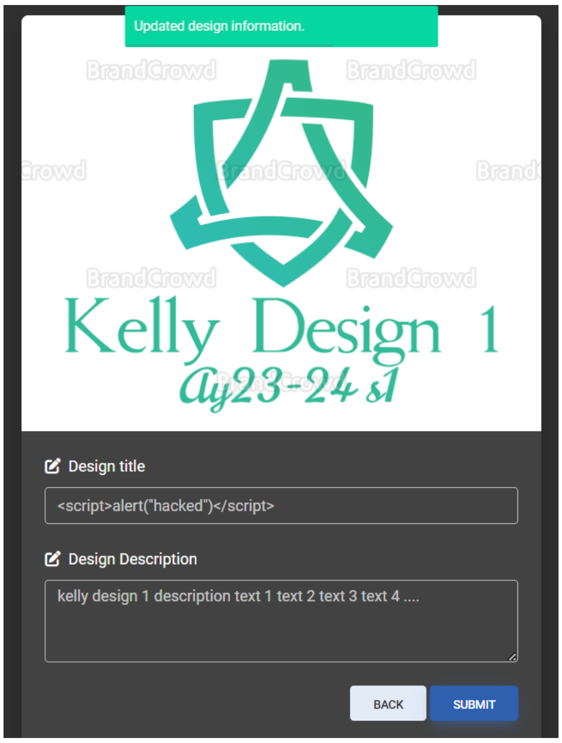
Design title

<script>alert("hacked")</script>

Design Description

kelly design 1 description text 1 text 2 text 3 text 4 ....

Step 4: Make sure the design information is updated


Updated design information.

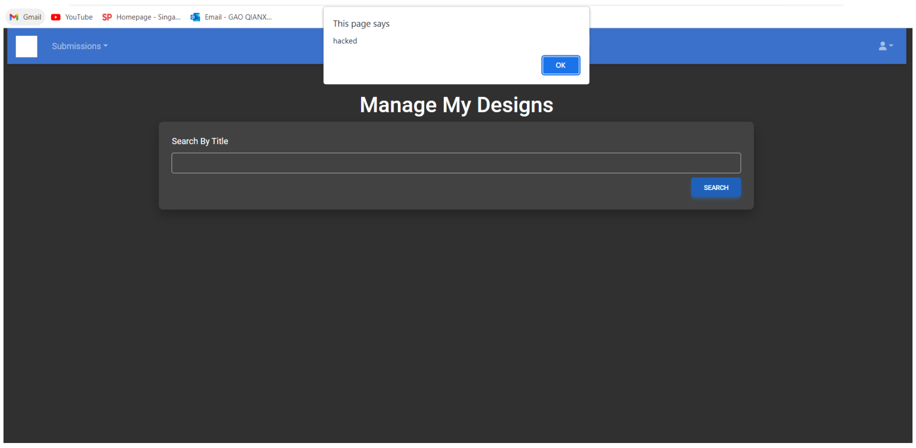
Design title

<script>alert("hacked")</script>

Design Description

kelly design 1 description text 1 text 2 text 3 text 4 ....

Step 5: Click “back” -> Click “search” -> Message prompt



This page says

hacked

1.3 Cause of vulnerability

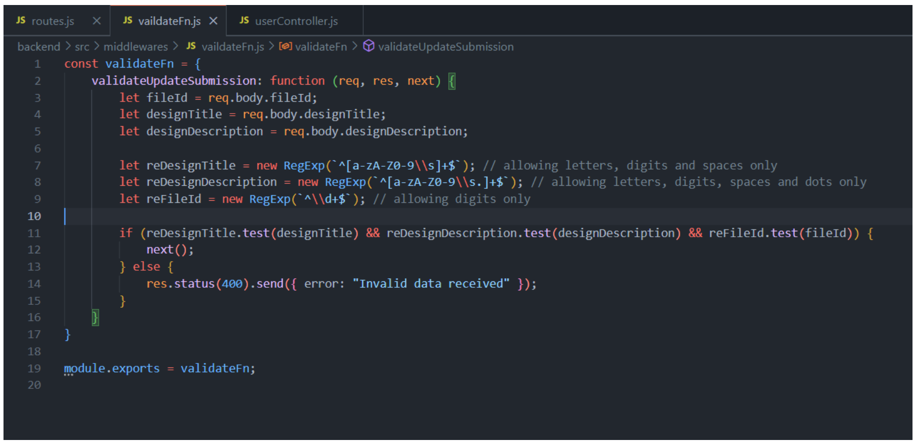
It does not have middleware for input validation.

```
router.put('/api/user/design/', userController.processUpdateOneDesign);
```

1.4 How to rectify vulnerability

I have created a file called “validateFn”.

I used a regular expression to validate the input.

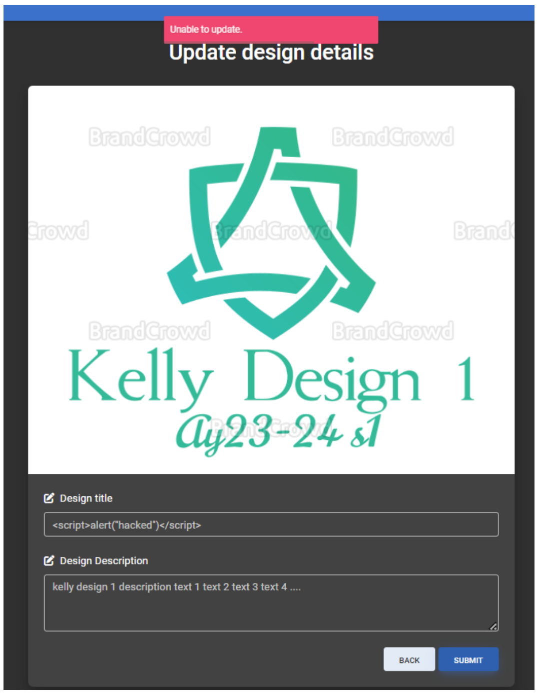

routes.js
vaildateFn.js
userController.js
backend > src > middlewares > vaildateFn.js > validateFn > validateUpdateSubmission

```
const validateFn = {
    validateUpdateSubmission: function (req, res, next) {
        let fileId = req.body.fileId;
        let designTitle = req.body.designTitle;
        let designDescription = req.body.designDescription;

        let reDesignTitle = new RegExp(`^[a-zA-Z0-9\\s]+$`); // allowing letters, digits and spaces only
        let reDesignDescription = new RegExp(`^[a-zA-Z0-9\\s.]+$`); // allowing letters, digits, spaces and dots only
        let reFileId = new RegExp(`^\\d+$`); // allowing digits only

        if (reDesignTitle.test(designTitle) && reDesignDescription.test(designDescription) && reFileId.test(fileId)) {
            next();
        } else {
            res.status(400).send({ error: "Invalid data received" });
        }
    }
}

module.exports = validateFn;
```

Include the file as middleware in the code.

```
router.put('/api/user/design/', validateFn.validateUpdateSubmission, userController.processUpdateOneDesign);
```

1.5 Testing


Unable to update.

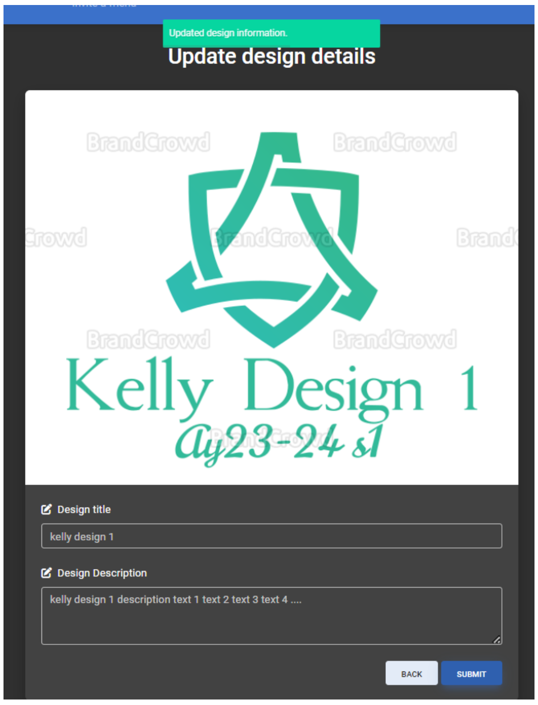
Update design details

Design title

<script>alert("hacked")</script>

Design Description

kelly design 1 description text 1 text 2 text 3 text 4 ....


Updated design information.

Update design details

Design title

kelly design 1

Design Description

kelly design 1 description text 1 text 2 text 3 text 4 ....

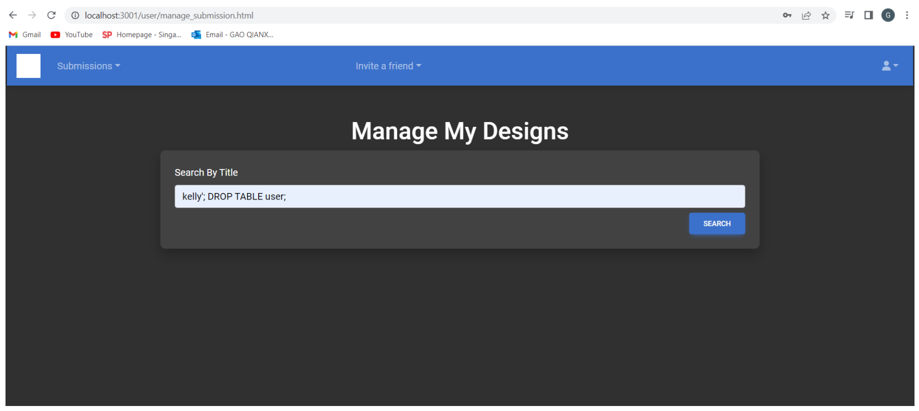
2. SQL Injection

2.1 Definition

It occurs when untrusted data is sent to an interpreter as part of a command or query, leading to the execution of unintended commands.

2.2 How to exploit vulnerability

Step 1: Login as kelly -> In the search by title textbox type "kelly'; DROP TABLE user;"

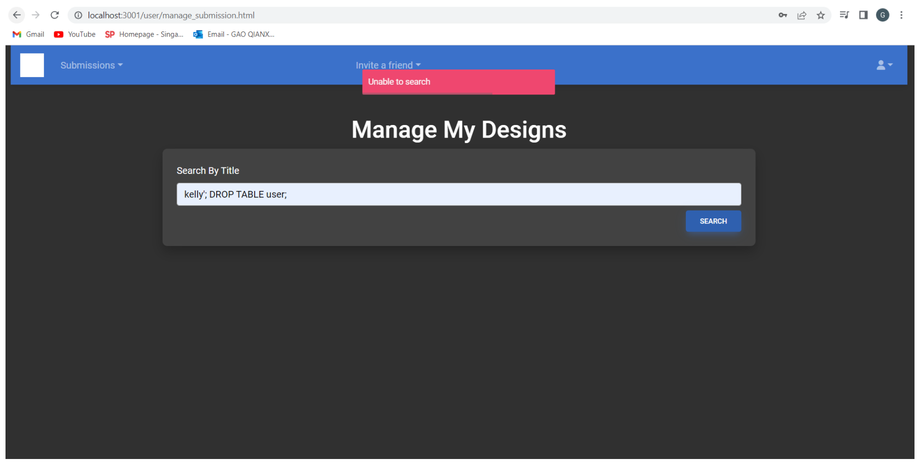


Manage My Designs

Search By Title

kelly'; DROP TABLE user;

Step 2: Click on "search"



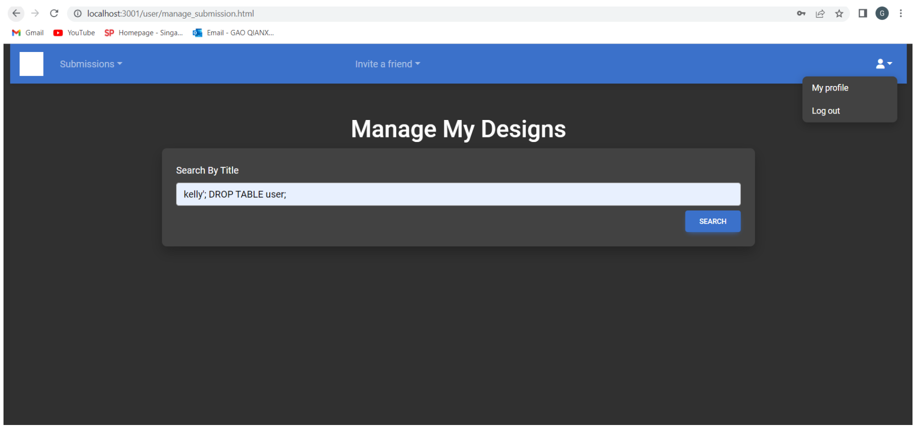
Unable to search

Manage My Designs

Search By Title

kelly'; DROP TABLE user;

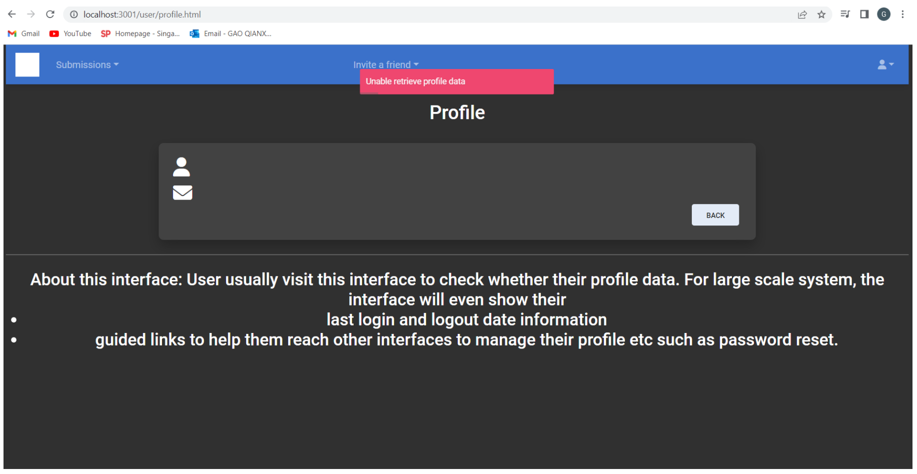
Step 3: Click on “My profile” (the profile is gone)



Manage My Designs

Search By Title

kelly'; DROP TABLE user;



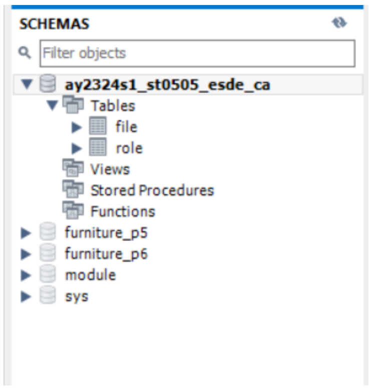
Unable retrieve profile data

Profile

About this interface: User usually visit this interface to check whether their profile data. For large scale system, the interface will even show their

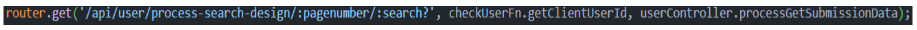
- last login and logout date information
- guided links to help them reach other interfaces to manage their profile etc such as password reset.

Step 5: Check the SQL workbench (the user table is being dropped)



2.3 Cause of vulnerability

It does not have middleware for input validation.



```
router.get('/api/user/process-search-design/:pagenumber/:search?', checkUserFn.getClientUserId, userController.processGetSubmissionData);
```

2.4 How to rectify vulnerability

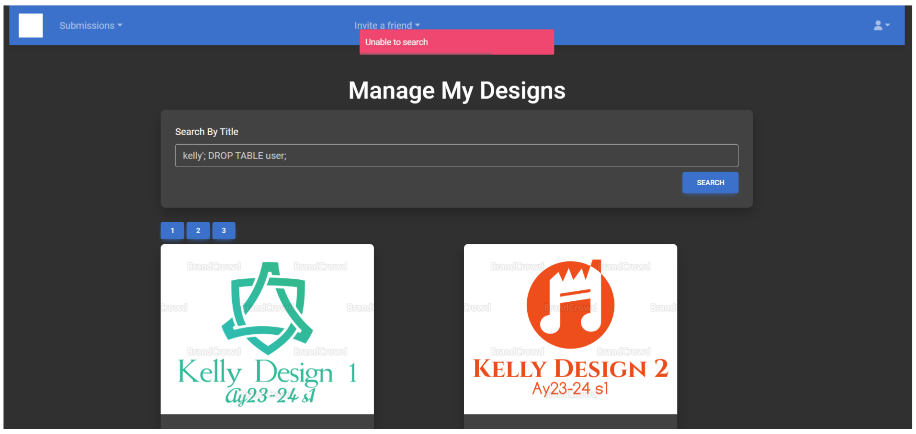
I used a regular expression to validate the input. (validateFn file)

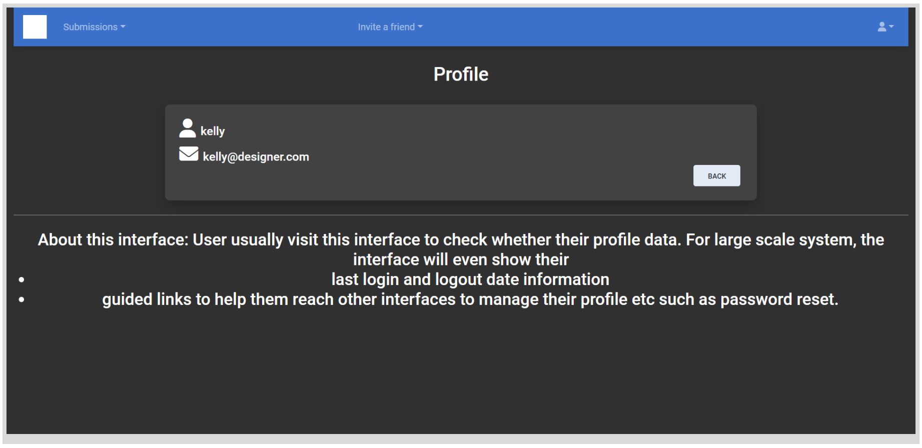
```
validateSubmissionDataInput: function (req, res, next) {
    let search = req.params.search;

    let reSearch = new RegExp(`^[a-zA-Z0-9\\s]+$`); // allowing letters, digits and spaces only

    if (reSearch.test(search)) {
        next();
    } else {
        res.status(400).send({ error: "Invalid data received" });
    }
}
```

Include the file as middleware in the code.

```
router.get('/api/user/process-search-design/:pagenumber/:search?', checkUserFn.getClientUserId, validateFn.validateSubmissionDataInput, userController.processGetSubmissionData);
```

2.5 Testing





About this interface: User usually visit this interface to check whether their profile data. For large scale system, the interface will even show their

- last login and logout date information
- guided links to help them reach other interfaces to manage their profile etc such as password reset.

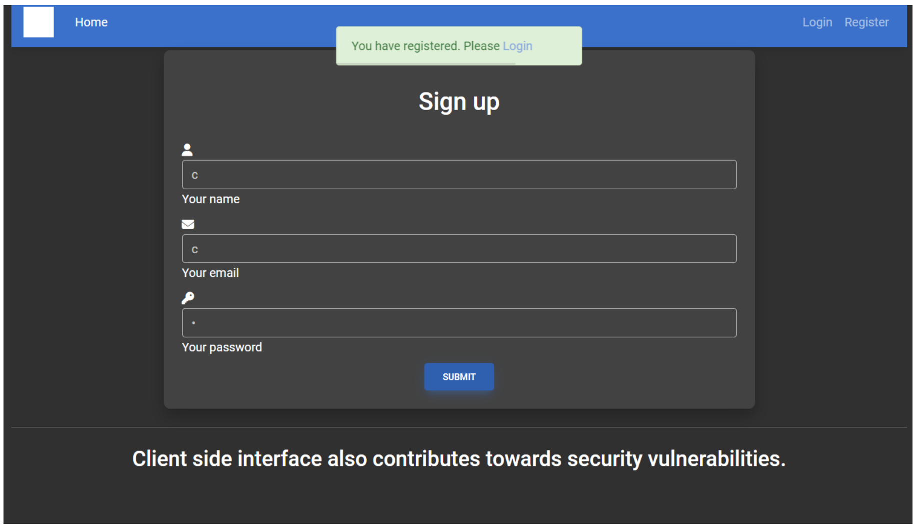
3. Broken Authentication

3.1 Definition

It occurs when credential verification or session management are poorly implemented, allowing attackers to compromise passwords or session tokens to assume users' identities.

3.2 How to exploit vulnerability

Step 1: Go to the register page, register by entering “c” for all -> click on “submit”



You have registered. Please Login

Sign up

c

Your name

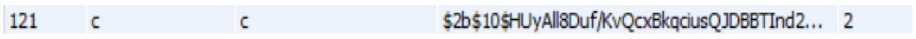
c

Your email

•

Your password

Client side interface also contributes towards security vulnerabilities.



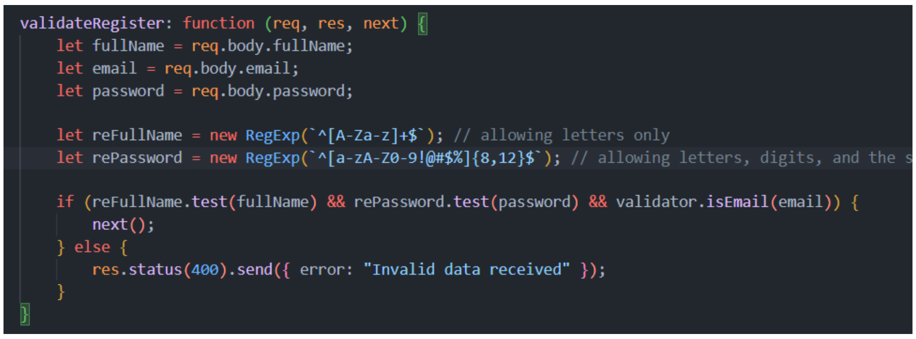
121	c	c	$2b$10$HUyAll8Duf/KvQcxBkqciusQJDBBTInd2...	2

3.3 Cause of vulnerability

There is no input validation for user registration.

3.4 How to rectify vulnerability

I utilized a regular expression to validate the user input during the registration process.



```
validateRegister: function (req, res, next) {
    let fullName = req.body.fullName;
    let email = req.body.email;
    let password = req.body.password;

    let reFullName = new RegExp(`^[A-Za-z]+$`); // allowing letters only
    let rePassword = new RegExp(`^[a-zA-Z0-9!@#$%]{8,12}$`); // allowing letters, digits, and the s

    if (reFullName.test(fullName) && rePassword.test(password) && validator.isEmail(email)) {
        next();
    } else {
        res.status(400).send({ error: "Invalid data received" });
    }
}
```

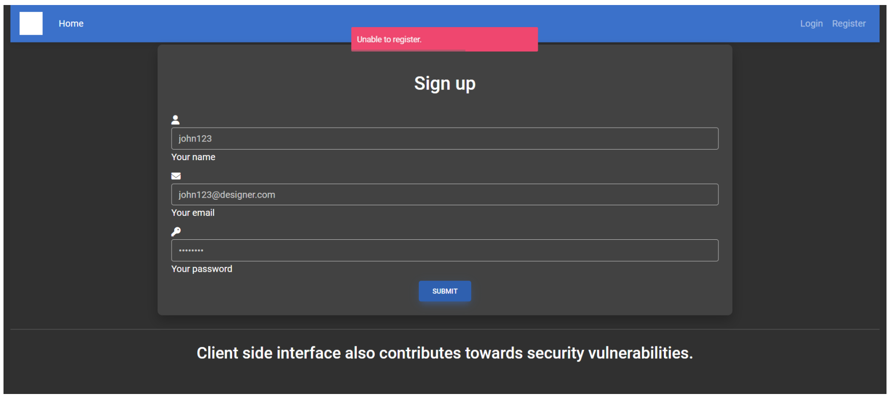
Include the file as middleware in the code.

```
router.post('/api/user/register',validateFn.validateRegister, authController.processRegister);
```

Additionally, change all user and admin passwords in the database. Make sure that the passwords are difficult to guess.

3.5 Testing

Password: password



Unable to register.

Sign up

john123

Your name

john123@designer.com

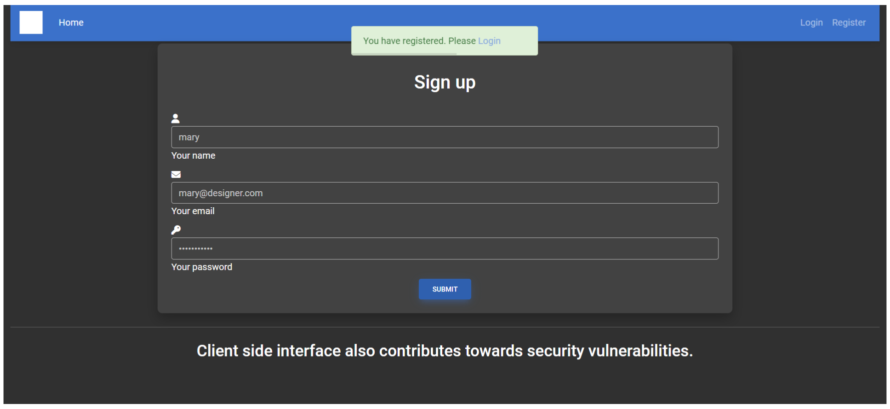
Your email

········

Your password

Client side interface also contributes towards security vulnerabilities.

Password: moN1pr#S@ap



You have registered. Please Login

Sign up

mary

Your name

mary@designer.com

Your email

···········

Your password

Client side interface also contributes towards security vulnerabilities.

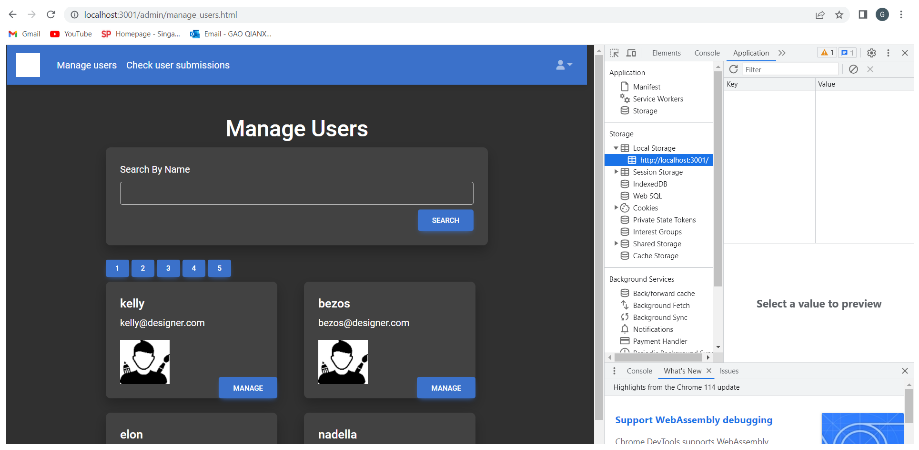
4. Broken Access Control

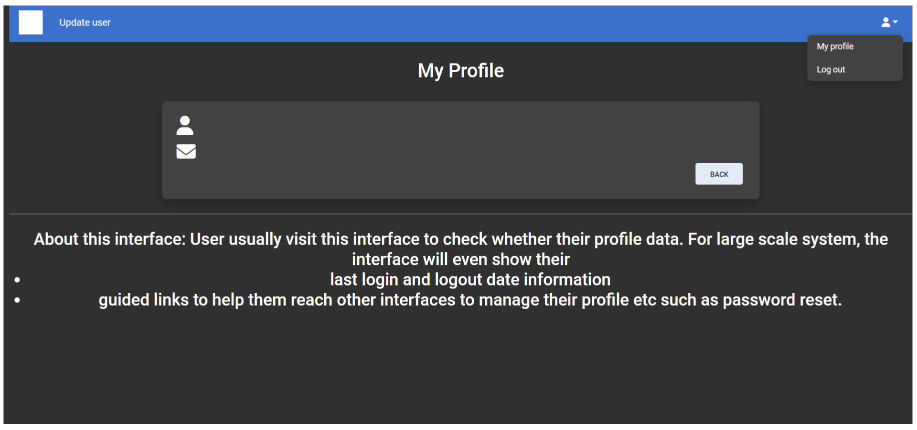
4.1 Definition

It occurs when permissions on resources are improperly enforced, allowing attackers to access unauthorized functionality or data.

4.2 How to exploit vulnerability

Step 1: Go to: http://localhost:3001/admin/manage_users.html

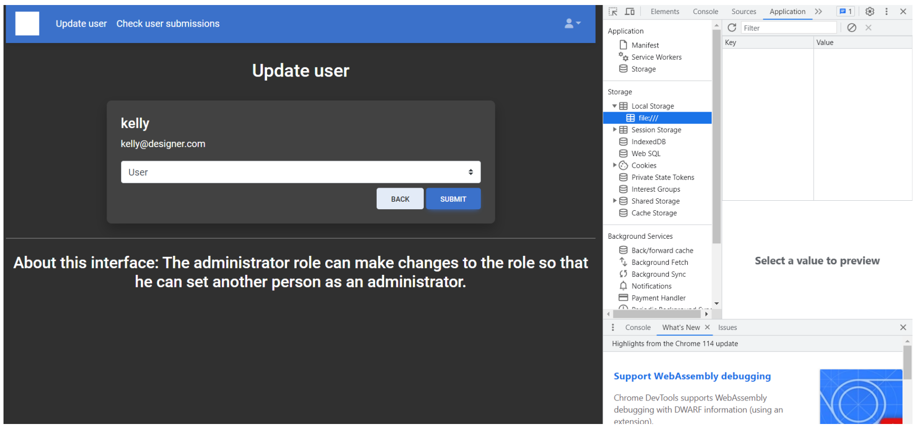




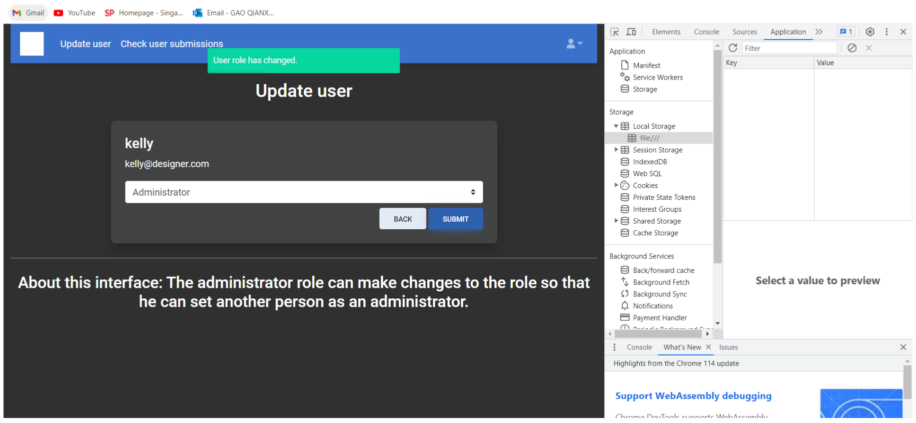
About this interface: User usually visit this interface to check whether their profile data. For large scale system, the interface will even show their

- last login and logout date information
- guided links to help them reach other interfaces to manage their profile etc such as password reset.

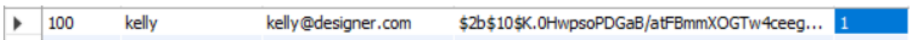
Step 2: On kelly click “Manage”


Update user
Check user submissions
Update user
kelly
kelly@designer.com
User
BACK
SUBMIT
About this interface: The administrator role can make changes to the role so that he can set another person as an administrator.

Step 3: Change kelly role from user to administrator


Update user
Check user submissions
User role has changed.
Update user
kelly
kelly@designer.com
Administrator
BACK
SUBMIT
About this interface: The administrator role can make changes to the role so that he can set another person as an administrator.

It has been changed



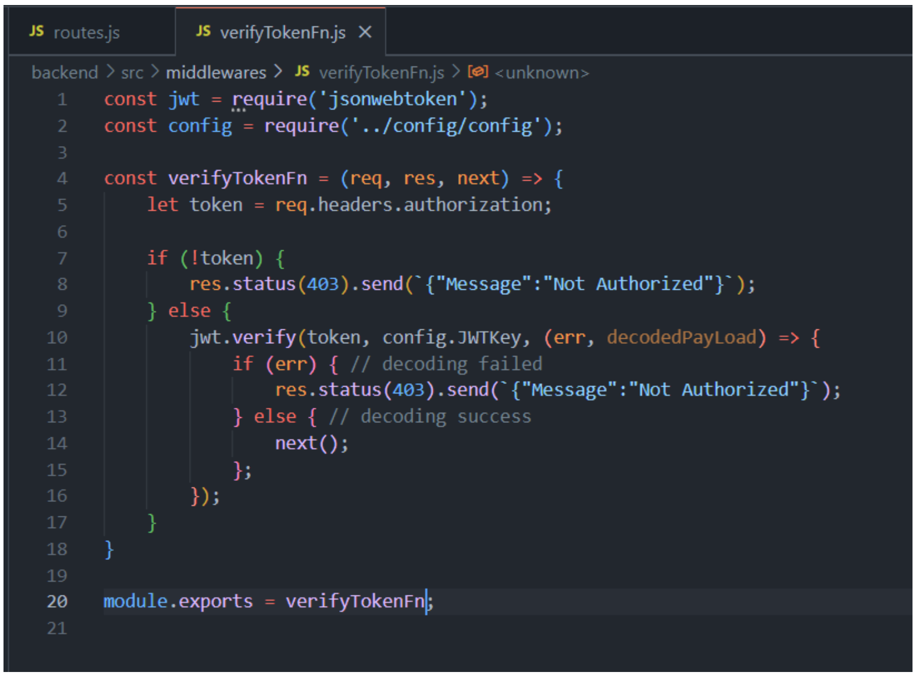
▶	100	kelly	kelly@designer.com	$2b$10$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg...	1

4.3 Cause of vulnerability

There is no middleware to block unauthorized access.

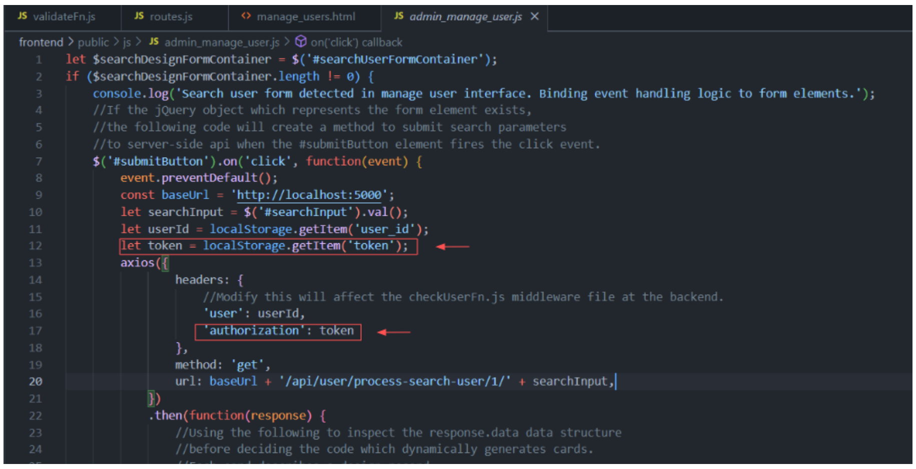
4.4 How to rectify vulnerability

I have created a file called “verifyTokenFn” to validate web token.

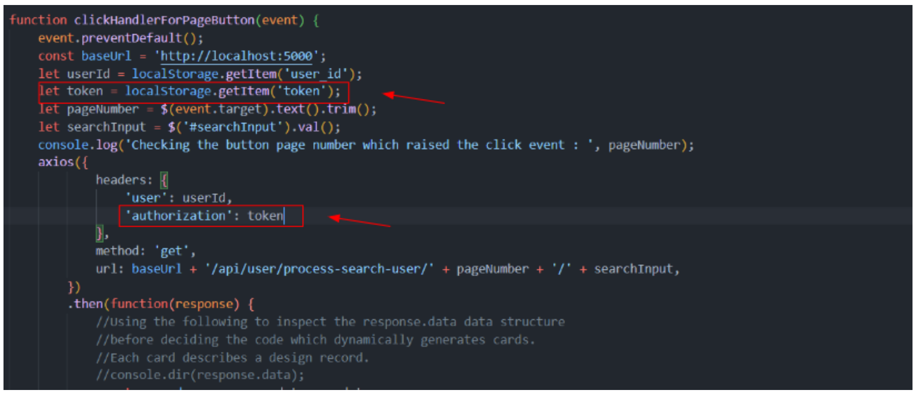


```
routes.js    verifyTokenFn.js
backend > src > middlewares > verifyTokenFn.js > <unknown>

const jwt = require('jsonwebtoken');
const config = require('../config/config');

const verifyTokenFn = (req, res, next) => {
    let token = req.headers.authorization;

    if (!token) {
        res.status(403).send(`{"Message":"Not Authorized"}`);
    } else {
        jwt.verify(token, config.JWTKey, (err, decodedPayLoad) => {
            if (err) { // decoding failed
                res.status(403).send(`{"Message":"Not Authorized"}`);
            } else { // decoding success
                next();
            };
        });
    }
}

module.exports = verifyTokenFn;
```

Under frontend -> public -> js -> admin_manage_user.js, add these codes

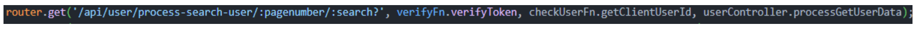


```
validateFn.js    routes.js    manage_users.html    admin_manage_user.js
frontend > public > js > admin_manage_user.js > on('click') callback

let $searchDesignFormContainer = $('#searchUserFormContainer');
if ($searchDesignFormContainer.length != 0) {
    console.log('Search user form detected in manage user interface. Binding event handling logic to form elements.');
    //If the jQuery object which represents the form element exists,
    //the following code will create a method to submit search parameters
    //to server-side api when the #submitButton element fires the click event.
    $('#submitButton').on('click', function(event) {
        event.preventDefault();
        const baseUrl = 'http://localhost:5000';
        let searchInput = $('#searchInput').val();
        let userId = localStorage.getItem('user_id');
        let token = localStorage.getItem('token');
        axios({
                headers: {
                    //Modify this will affect the checkUserFn.js middleware file at the backend.
                    'user': userId,
                    'authorization': token
                },
                method: 'get',
                url: baseUrl + '/api/user/process-search-user/1/' + searchInput,
            })
            .then(function(response) {
                //Using the following to inspect the response.data data structure
                //before deciding the code which dynamically generates cards.
```



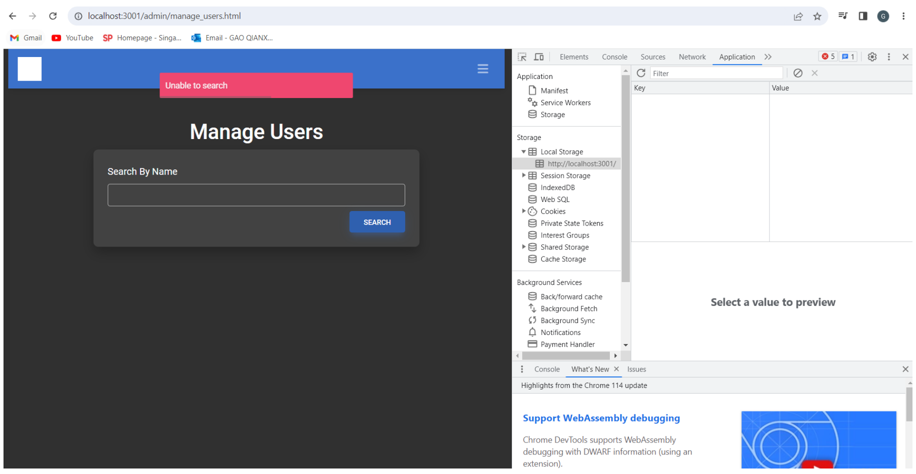
```
function clickHandlerForPageButton(event) {
    event.preventDefault();
    const baseUrl = 'http://localhost:5000';
    let userId = localStorage.getItem('user_id');
    let token = localStorage.getItem('token');
    let pageNumber = $(event.target).text().trim();
    let searchInput = $('#searchInput').val();
    console.log('Checking the button page number which raised the click event : ', pageNumber);
    axios({
            headers: {
                'user': userId,
                'authorization': token
            },
            method: 'get',
            url: baseUrl + '/api/user/process-search-user/' + pageNumber + '/' + searchInput,
        })
        .then(function(response) {
            //Using the following to inspect the response.data data structure
            //before deciding the code which dynamically generates cards.
            //Each card describes a design record.
            //console.dir(response.data);
```

Include the file as middleware in the code.



```
router.get('/api/user/process-search-user/:pagenumber/:search?', verifyFn.verifyToken, checkUserFn.getClientUserId, userController.processGetUserData);
```

4.5 Testing



localhost:3001/admin/manage_users.html

Unable to search

Manage Users

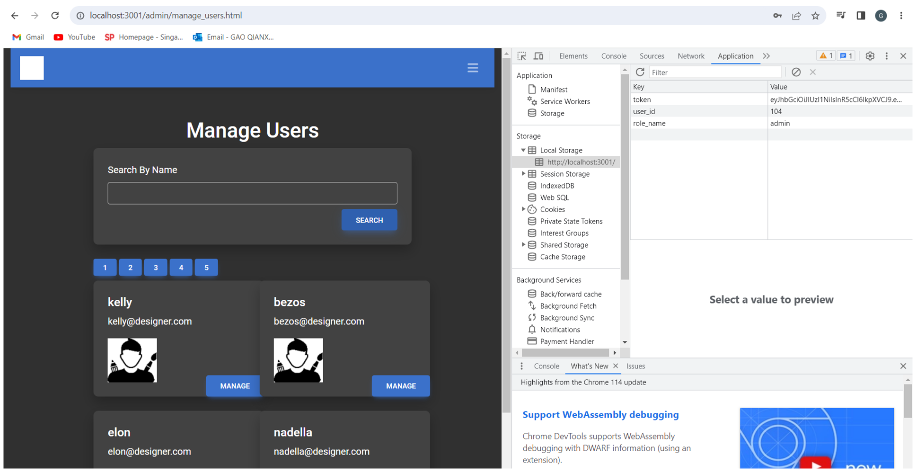
Search By Name

SEARCH

Highlights from the Chrome 114 update

Support WebAssembly debugging

Chrome DevTools supports WebAssembly debugging with DWARF information (using an extension).



localhost:3001/admin/manage_users.html

Manage Users

Search By Name

SEARCH

1 2 3 4 5

kelly
kelly@designer.com
MANAGE

bezos
bezos@designer.com
MANAGE

elon
elon@designer.com

nadella
nadella@designer.com

Key	Value
token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e...
user_id	104
role_name	admin

Highlights from the Chrome 114 update

Support WebAssembly debugging

Chrome DevTools supports WebAssembly debugging with DWARF information (using an extension).

5. Sensitive Data Exposure

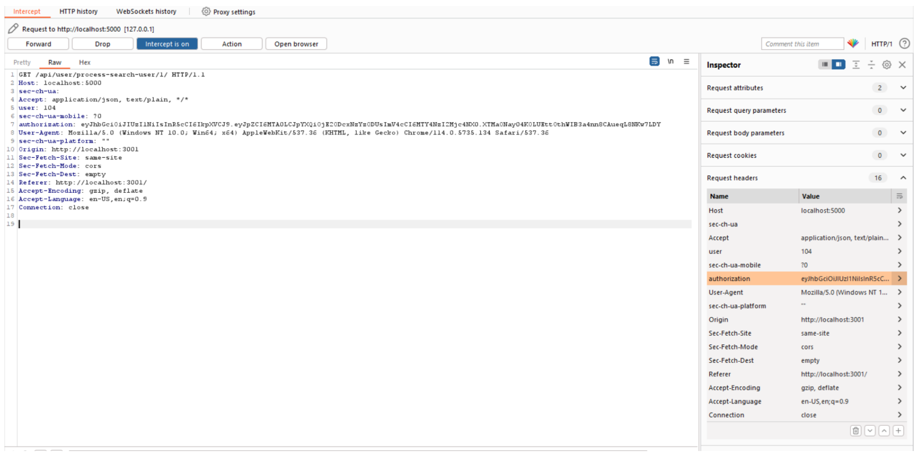
5.1 Definition

It occurs due to lack or weak protection of sensitive data, allowing attackers to steal or modify the data when the data is at rest or in transit.

5.2 How to exploit vulnerability

Step 1: Open Burp Suite -> go to proxy -> under intercept, open browser and enter http://localhost:3001

Step 2: Login as Albert -> go back to burp suite -> keep pressing "forward" until this appears



Request to http://localhost:5000 [127.0.0.1]

```
GET /api/user/process-search-user/1/ HTTP/1.1
Host: localhost:5000
sec-ch-ua:
Accept: application/json, text/plain, */*
user: 104
sec-ch-ua-mobile: ?0
authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTA0LCJpYXQiOjE2OTcxNzYzODUsImV4cCI6MTY4NzI2Mjc4NX0.XTMa0Nay04K0LUEtt0thWIB3a4nn8CAueqL8NKw7LDY
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.134 Safari/537.36
sec-ch-ua-platform: ""
Origin: http://localhost:3001
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3001/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

```

Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 0
Request headers 16

Name	Value
Host	localhost:5000
sec-ch-ua	
Accept	application/json, text/plain...
user	104
sec-ch-ua-mobile	?0
authorization	eyJhbGciOiJIUzI1NiIsInR5cC...
User-Agent	Mozilla/5.0 (Windows NT 1...
sec-ch-ua-platform	""
Origin	http://localhost:3001
Sec-Fetch-Site	same-site
Sec-Fetch-Mode	cors
Sec-Fetch-Dest	empty
Referer	http://localhost:3001/
Accept-Encoding	gzip, deflate
Accept-Language	en-US,en;q=0.9
Connection	close

5.3 Cause of vulnerability

The website link is using insecure HTTP protocol.

5.4 How to rectify vulnerability

Step 1: Open command prompt using administrator, Install openSSL using chocolaty:

- choco -v (check chocolaty version to make sure it is installed)
- choco install openssl.light (install openSSL)
- openssl version (check openSSL version to make sure it is installed)

Step 3: Generate a private key:

Run this command, it generates a 2048 bit RSA private key and saves it in the ‘privateKey.key’ file -> “openssl genpkey -algorithm RSA -out privateKey.key -pkeyopt rsa_keygen_bits:2048“

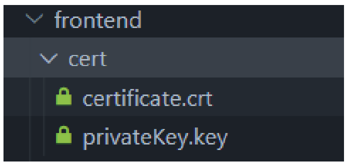
Step 4: Generate a certificate signing request:

Run this command -> “openssl req -new -key privateKey.key -out csr.csr”

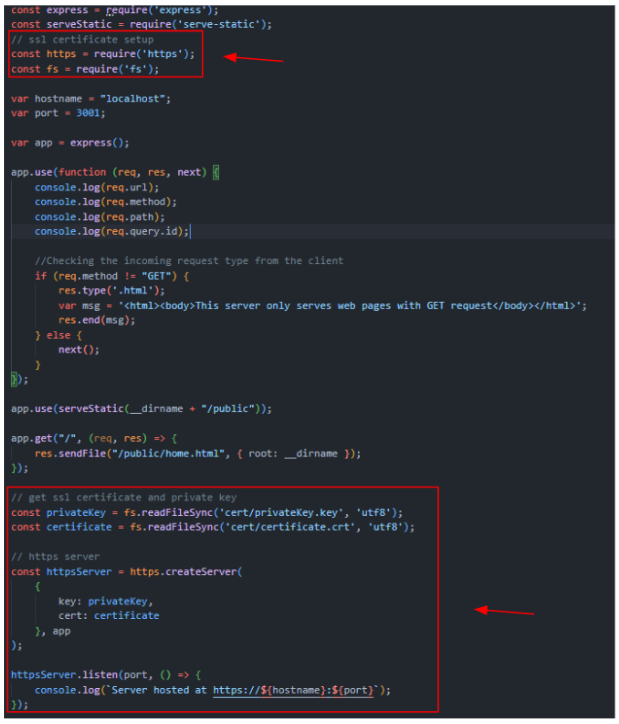
Step 5: Generate a self-signed certificate:

Run this command, it creates a self-signed certificate valid for 365 days and saves it in the ‘certificate.crt’ file -> “openssl x509 -req -days 365 -in csr.csr -signkey privateKey.key -out certificate.crt”

Step 6: Open vs code -> frontend -> create a new file called “cert” -> put in the privateKey.key and certificate.crt files



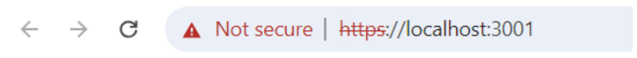
Step 7: open frontend, index.js -> add in these codes



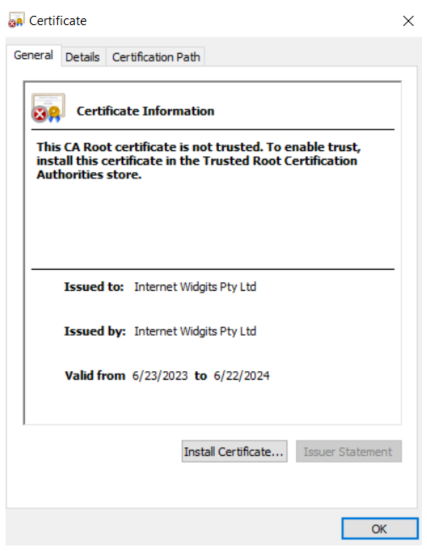
```javascript
const express = require('express');
const serveStatic = require('serve-static');
// ssl certificate setup
const https = require('https');
const fs = require('fs');

var hostname = "localhost";
var port = 3001;

var app = express();

app.use(function (req, res, next) {
    console.log(req.url);
    console.log(req.method);
    console.log(req.path);
    console.log(req.query.id);

    //Checking the incoming request type from the client
    if (req.method != "GET") {
        res.type('.html');
        var msg = '<html><body>This server only serves web pages with GET request</body></html>';
        res.end(msg);
    } else {
        next();
    }
});

app.use(serveStatic(__dirname + "/public"));

app.get("/", (req, res) => {
    res.sendFile("/public/home.html", { root: __dirname });
});

// get ssl certificate and private key
const privateKey = fs.readFileSync('cert/privateKey.key', 'utf8');
const certificate = fs.readFileSync('cert/certificate.crt', 'utf8');

// https server
const httpsServer = https.createServer(
    {
        key: privateKey,
        cert: certificate
    }, app
);

httpsServer.listen(port, () => {
    console.log(`Server hosted at https://${hostname}:${port}`);
});
```

5.5 Testing

It is fine that the browser says that the certificate used is “Not secure” because it warns us that the certificate is not recognized by the CA (Certification Authorities), but it is still a **valid** certificate.



Not secure | ~~https~~://localhost:3001



Certificate

General | Details | Certification Path

Certificate Information

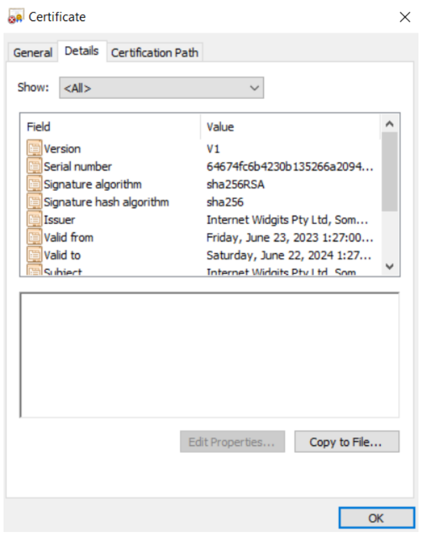
This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: Internet Widgits Pty Ltd

Issued by: Internet Widgits Pty Ltd

Valid from 6/23/2023 **to** 6/22/2024

Install Certificate... | Issuer Statement | OK



Certificate

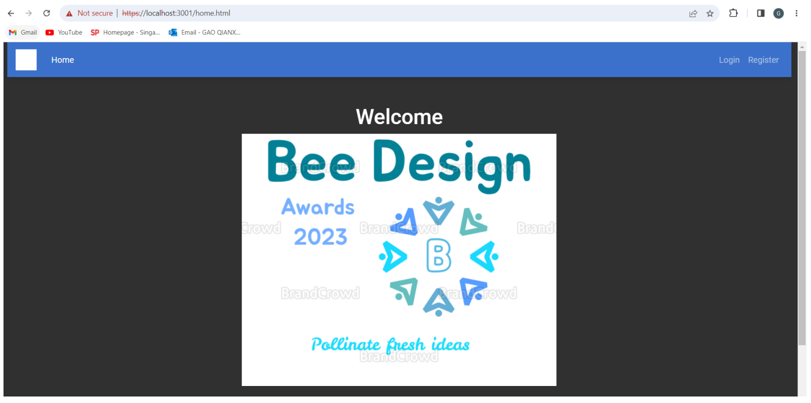
General | Details | Certification Path

Show: <All>

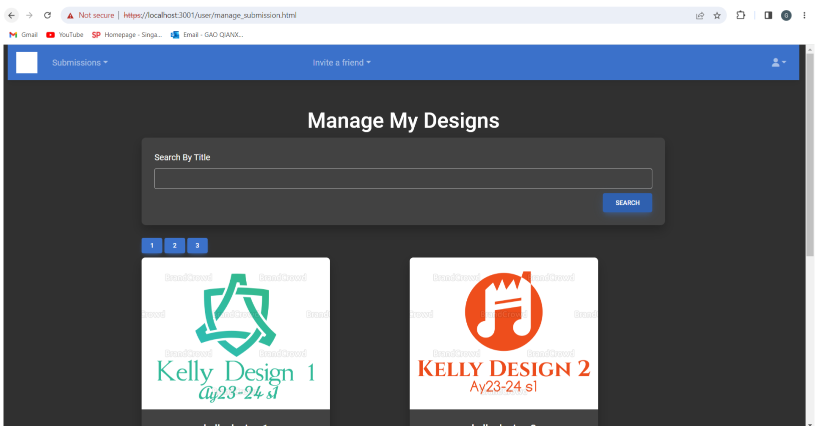
Field	Value
Version	V1
Serial number	64674fc6b4230b135266a2094...
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	Internet Widgits Pty Ltd, Som...
Valid from	Friday, June 23, 2023 1:27:00...
Valid to	Saturday, June 22, 2024 1:27:...
Subject	Internet Widgits Pty Ltd, Som...

Edit Properties... | Copy to File... | OK

All the pages are working well after changing http to https.



Not secure | ~~https~~://localhost:3001/home.html



Not secure | ~~https~~://localhost:3001/user/manage_submission.html

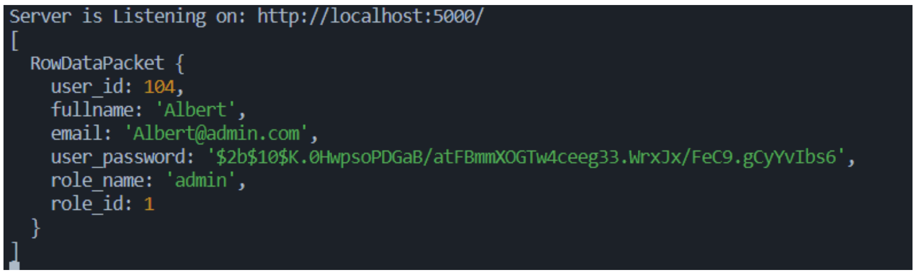
6. Insufficient Logging & Monitoring

6.1 Definition

It occurs when organizations do not put in-place robust logging mechanisms or respond to alerts of potential attacks, allowing attackers to achieve their goals without being detected.

6.2 How to exploit vulnerability

I realized that the backend does not have enough monitoring.



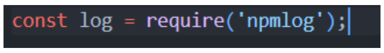
```
Server is Listening on: http://localhost:5000/
[
  RowDataPacket {
    user_id: 104,
    fullname: 'Albert',
    email: 'Albert@admin.com',
    user_password: '$2b$10$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6',
    role_name: 'admin',
    role_id: 1
  }
]
```

6.3 Cause of vulnerability

Insufficient Logging & Monitoring.

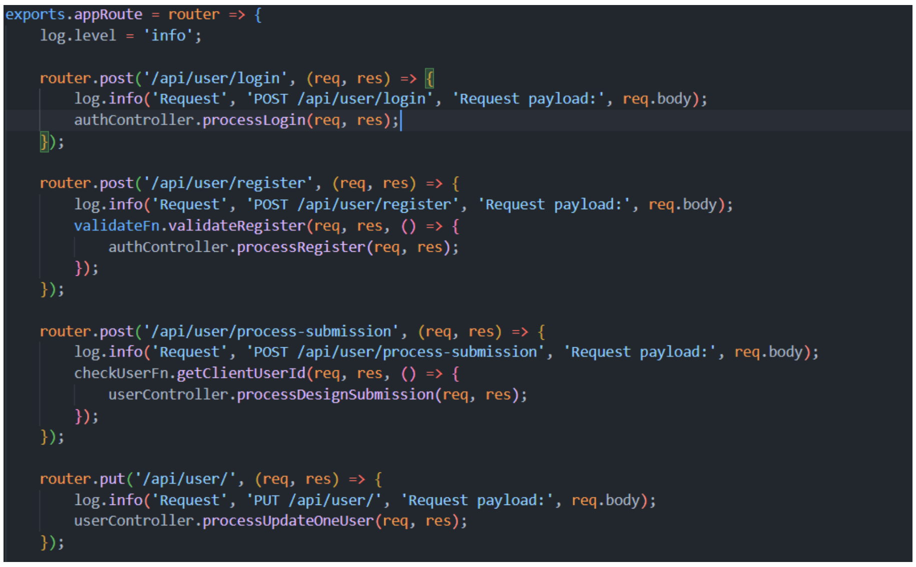
6.4 How to rectify vulnerability

Step 1: Go to backend -> routes.js -> add in this code

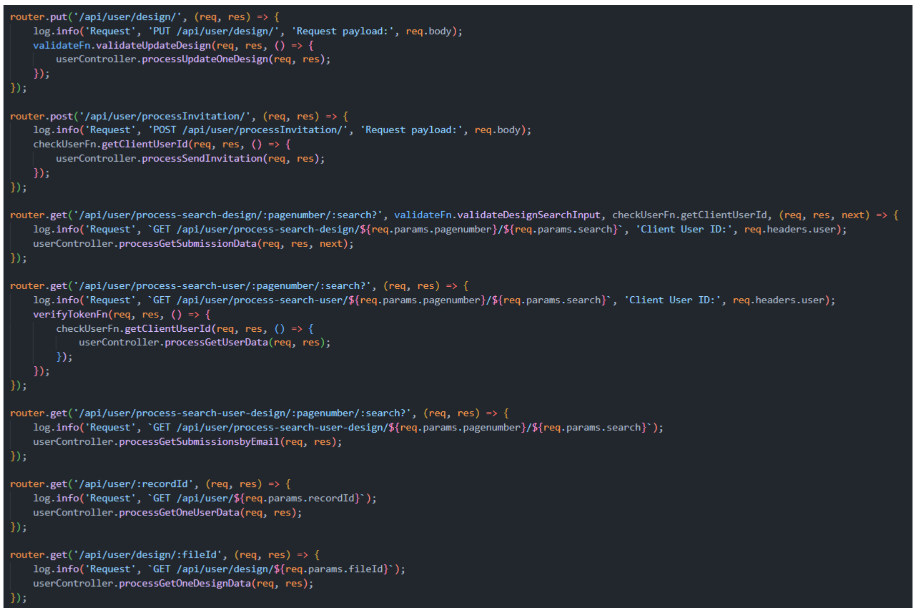


```
const log = require('npmlog');
```

Step 2: Replace the old codes with the following codes



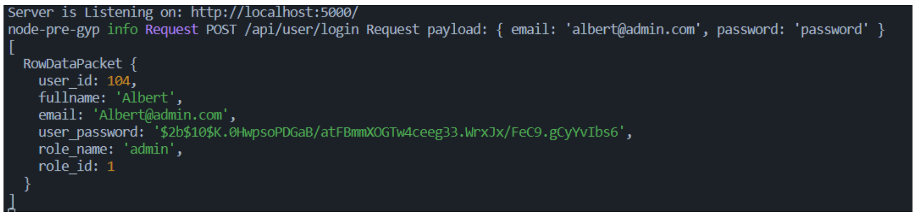
```
exports.appRoute = router => {
    log.level = 'info';

    router.post('/api/user/login', (req, res) => {
        log.info('Request', 'POST /api/user/login', 'Request payload:', req.body);
        authController.processLogin(req, res);
    });

    router.post('/api/user/register', (req, res) => {
        log.info('Request', 'POST /api/user/register', 'Request payload:', req.body);
        validateFn.validateRegister(req, res, () => {
            authController.processRegister(req, res);
        });
    });

    router.post('/api/user/process-submission', (req, res) => {
        log.info('Request', 'POST /api/user/process-submission', 'Request payload:', req.body);
        checkUserFn.getClientUserId(req, res, () => {
            userController.processDesignSubmission(req, res);
        });
    });

    router.put('/api/user/', (req, res) => {
        log.info('Request', 'PUT /api/user/', 'Request payload:', req.body);
        userController.processUpdateOneUser(req, res);
    });
```



```
router.put('/api/user/design/', (req, res) => {
    log.info('Request', 'PUT /api/user/design/', 'Request payload:', req.body);
    validateFn.validateUpdateDesign(req, res, () => {
        userController.processUpdateOneDesign(req, res);
    });
});

router.post('/api/user/processInvitation/', (req, res) => {
    log.info('Request', 'POST /api/user/processInvitation/', 'Request payload:', req.body);
    checkUserFn.getClientUserId(req, res, () => {
        userController.processSendInvitation(req, res);
    });
});

router.get('/api/user/process-search-design/:pagenumber/:search?', validateFn.validateDesignSearchInput, checkUserFn.getClientUserId, (req, res, next) => {
    log.info('Request', `GET /api/user/process-search-design/${req.params.pagenumber}/${req.params.search}`, 'Client User ID:', req.headers.user);
    userController.processGetSubmissionData(req, res, next);
});

router.get('/api/user/process-search-user/:pagenumber/:search?', (req, res) => {
    log.info('Request', `GET /api/user/process-search-user/${req.params.pagenumber}/${req.params.search}`, 'Client User ID:', req.headers.user);
    verifyTokenFn(req, res, () => {
        checkUserFn.getClientUserId(req, res, () => {
            userController.processGetUserData(req, res);
        });
    });
});

router.get('/api/user/process-search-user-design/:pagenumber/:search?', (req, res) => {
    log.info('Request', `GET /api/user/process-search-user-design/${req.params.pagenumber}/${req.params.search}`);
    userController.processGetSubmissionsbyEmail(req, res);
});

router.get('/api/user/:recordId', (req, res) => {
    log.info('Request', `GET /api/user/${req.params.recordId}`);
    userController.processGetOneUserData(req, res);
});

router.get('/api/user/design/:fileId', (req, res) => {
    log.info('Request', `GET /api/user/design/${req.params.fileId}`);
    userController.processGetOneDesignData(req, res);
});
```

6.5 Testing

The process is being monitored.



```
Server is Listening on: http://localhost:5000/
node-pre-gyp info Request POST /api/user/login Request payload: { email: 'albert@admin.com', password: 'password' }
[
  RowDataPacket {
    user_id: 104,
    fullname: 'Albert',
    email: 'Albert@admin.com',
    user_password: '$2b$10$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCyYvIbs6',
    role_name: 'admin',
    role_id: 1
  }
]
```