ST0505: SECURE CODING

Enterprise Development System

DIT/FT/2A/23

Done By:
Cheryl Ee 1949898 | Maylyn Yap 1838718

TABLE OF CONTENTS

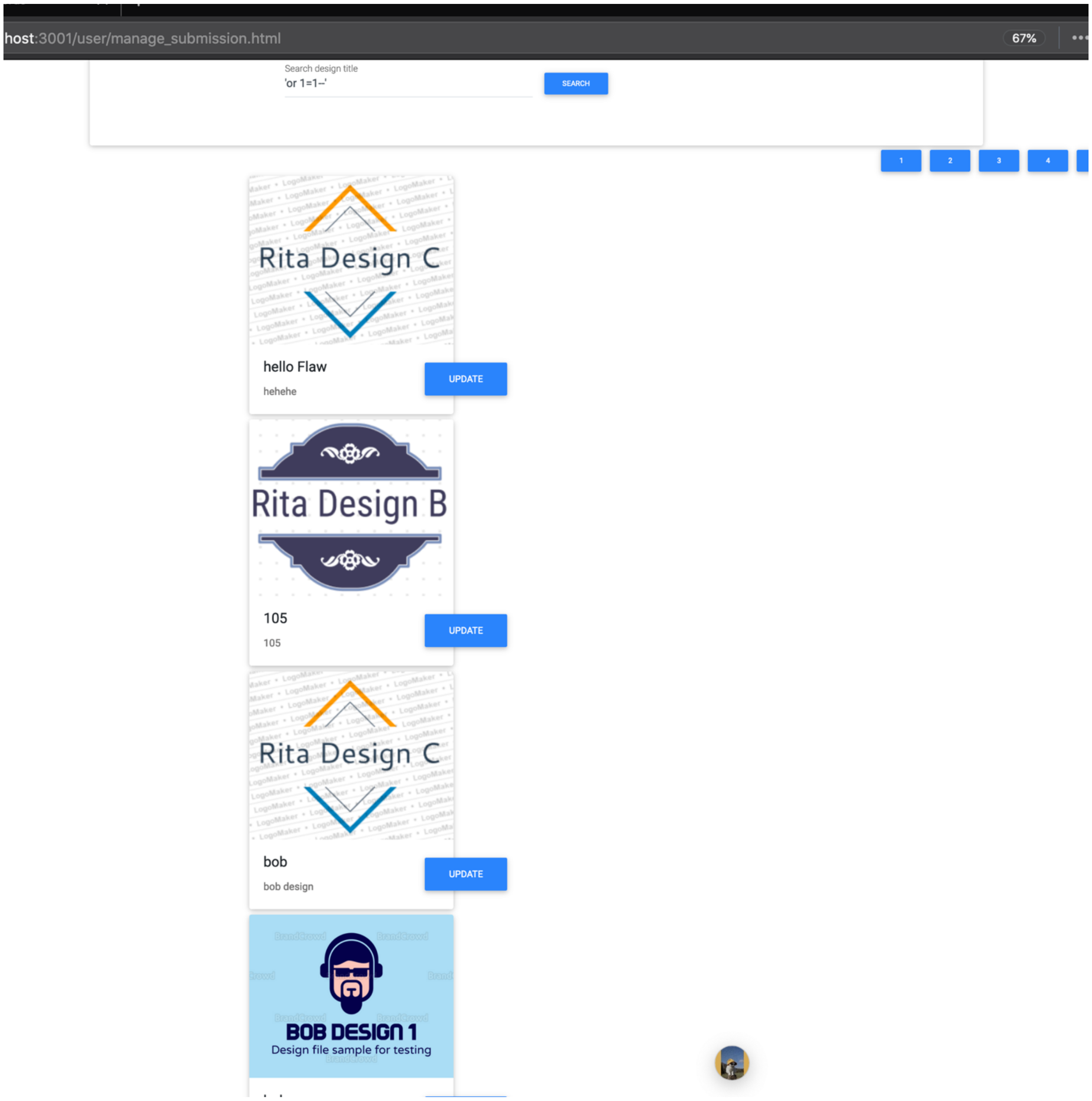
OVERVIEW

We have done an vulnerability assessment, evaluating potential security risks in [D]esign [C]ompetition to reduce the possibility pf threats. We have identified the potential vulnerabilities and have come up with a plausible solution and recommendations for your application.

SQL Injection

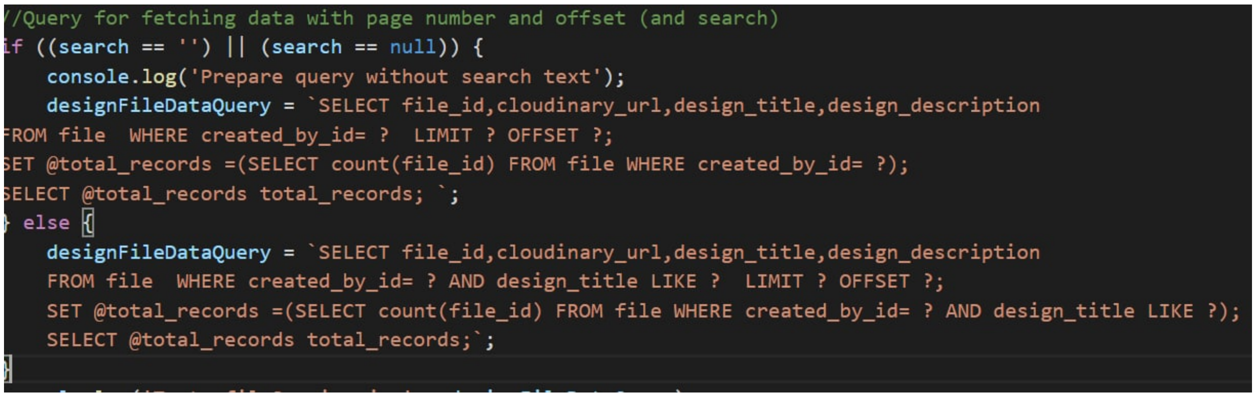
Injection can result in data loss, corruption or disclosure to unauthorized parties.

- **Vulnerability level**: High
- **Explanation**:
    - Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter.
    - Hostile data is directly used or concatenated, such that the SQL or command contains both structure and hostile data in dynamic queries, commands, or stored procedures.
- **Detailed Example**
    - **Signed as Rita but able to see Bob’s and other designers design submission**



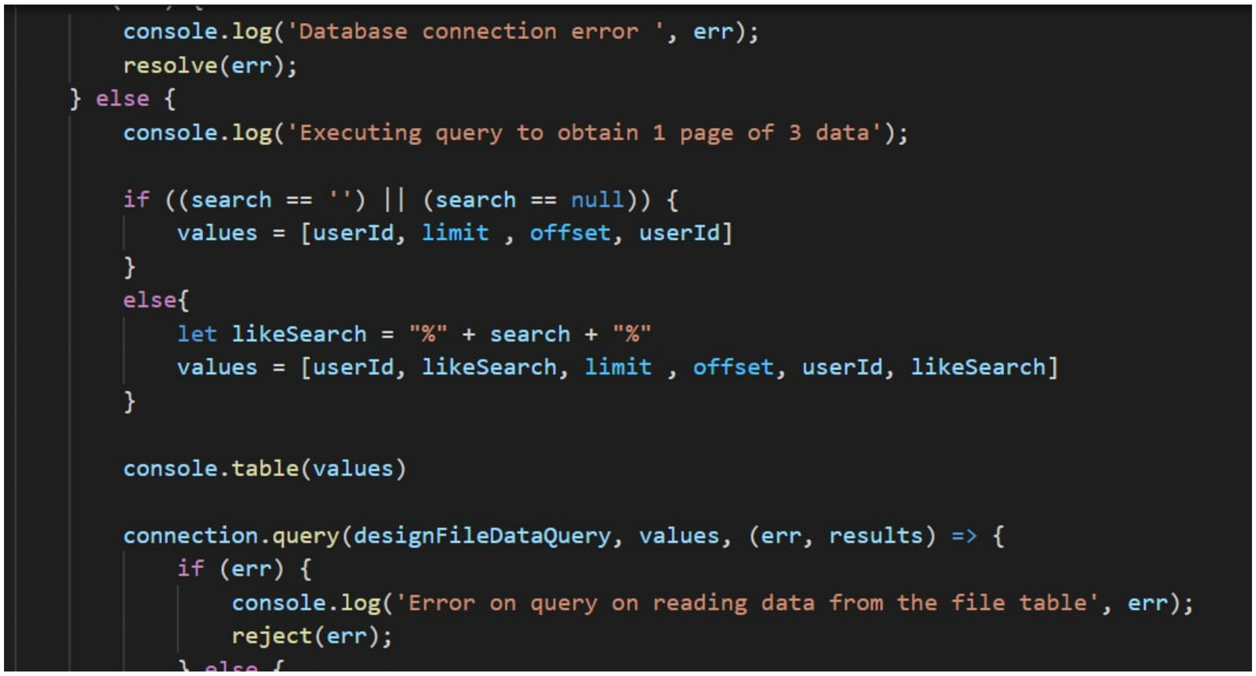
- **Recommendations**
    - Code should use positive or “whitelist” server-side input validation to keep data separate from commands and queries.

- **Solution**
    - **The query being executed should be changed to the following:**



```
//Query for fetching data with page number and offset (and search)
if ((search == '') || (search == null)) {
    console.log('Prepare query without search text');
    designFileDataQuery = `SELECT file_id,cloudinary_url,design_title,design_description
FROM file  WHERE created_by_id= ?  LIMIT ? OFFSET ?;
SET @total_records =(SELECT count(file_id) FROM file WHERE created_by_id= ?);
SELECT @total_records total_records; `;
} else {
    designFileDataQuery = `SELECT file_id,cloudinary_url,design_title,design_description
    FROM file  WHERE created_by_id= ? AND design_title LIKE ?  LIMIT ? OFFSET ?;
    SET @total_records =(SELECT count(file_id) FROM file WHERE created_by_id= ? AND design_title LIKE ?);
    SELECT @total_records total_records;`;
}
```

The above code is changed all the parameters that have ${} to ? (parametized query) , Data from '?' will be properly escaped and library function will parse the parameter.



```
        console.log('Database connection error ', err);
        resolve(err);
    } else {
        console.log('Executing query to obtain 1 page of 3 data');

        if ((search == '') || (search == null)) {
            values = [userId, limit , offset, userId]
        }
        else{
            let likeSearch = "%" + search + "%"
            values = [userId, likeSearch, limit , offset, userId, likeSearch]
        }

        console.table(values)

        connection.query(designFileDataQuery, values, (err, results) => {
            if (err) {
                console.log('Error on query on reading data from the file table', err);
                reject(err);
```

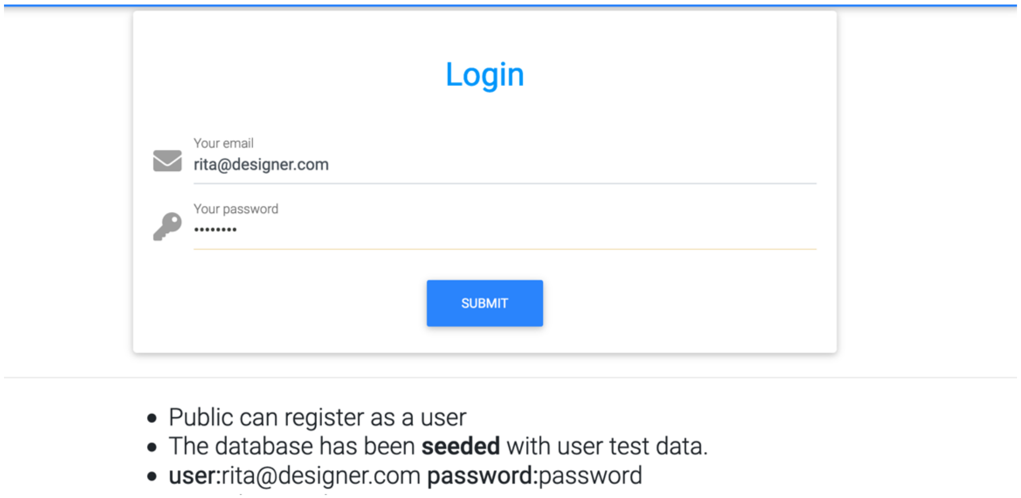
Broken Authentication

Attackers have to gain access to only a few accounts or admin account to compromise the system. This may allow identity theft or disclose legally protected highly sensitive information.

- **Vulnerability level**: High
- **Explanation**: Attackers are able to gain access as when user uses a public computer to access [D]esign [C]ompetition application, instead of selecting "logout" the user simply closes the browser tab and walks away, in

a result, attacker uses the same computer an hour later, user is still authenticated. Application session timeouts aren't set properly hence this is a broken authentication vulnerability.

- **Detailed Example**
  - **Login as Rita**



Login

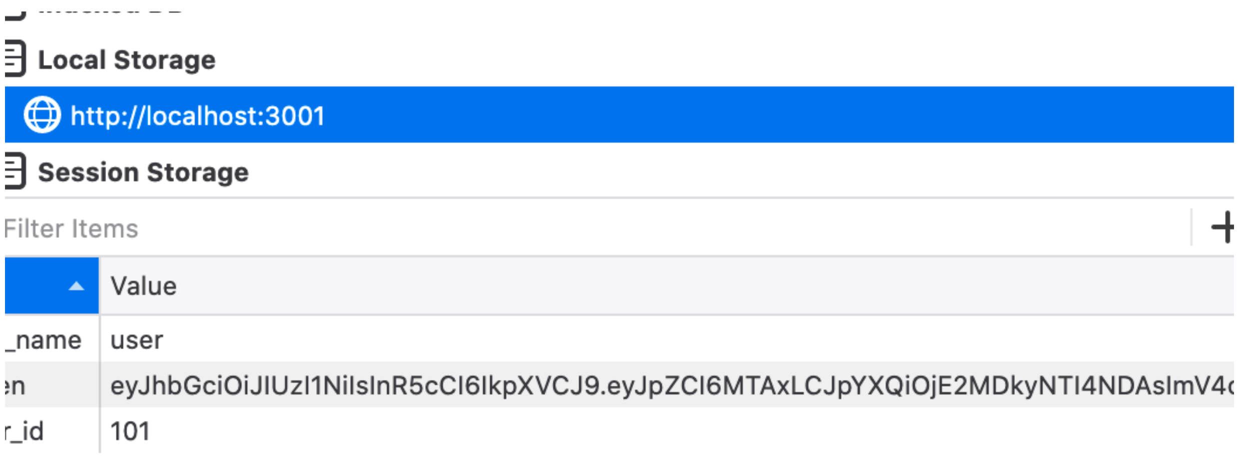
Your email
rita@designer.com

Your password
••••••••

SUBMIT

- Public can register as a user
- The database has been **seeded** with user test data.
- **user:**rita@designer.com **password:**password

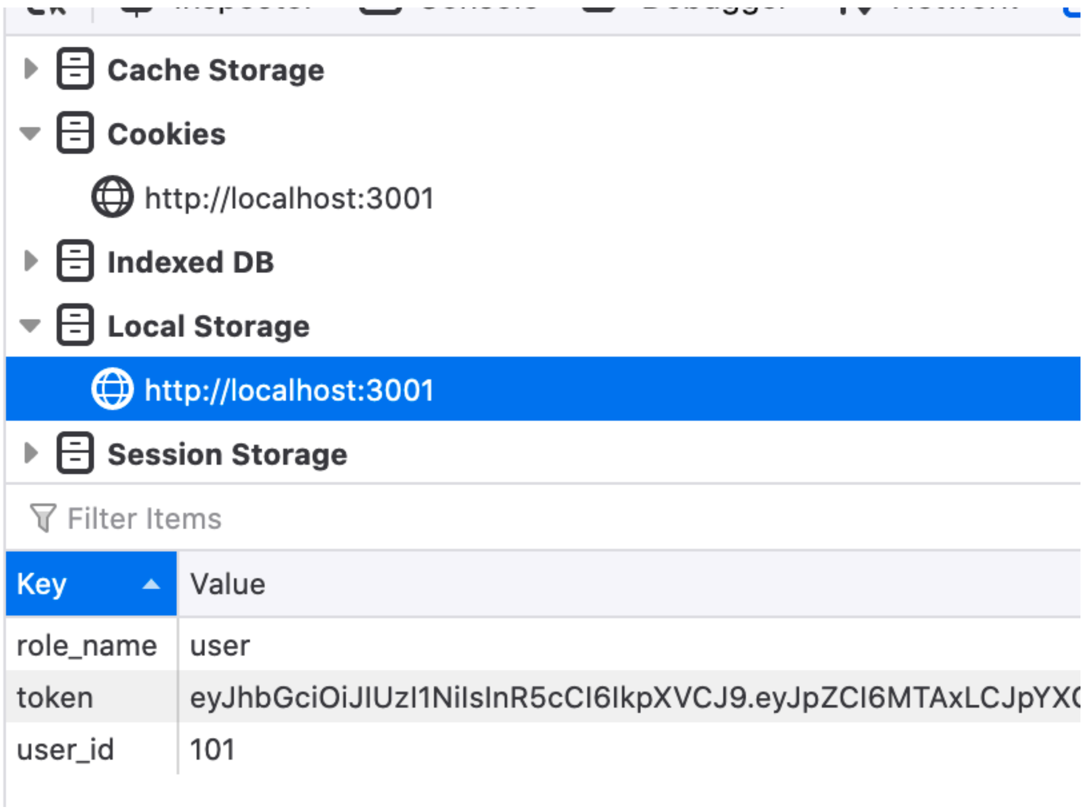
  - **Inspect element and go to Storage to view token to verify its Rita session Id**



Local Storage
http://localhost:3001
Session Storage
Filter Items

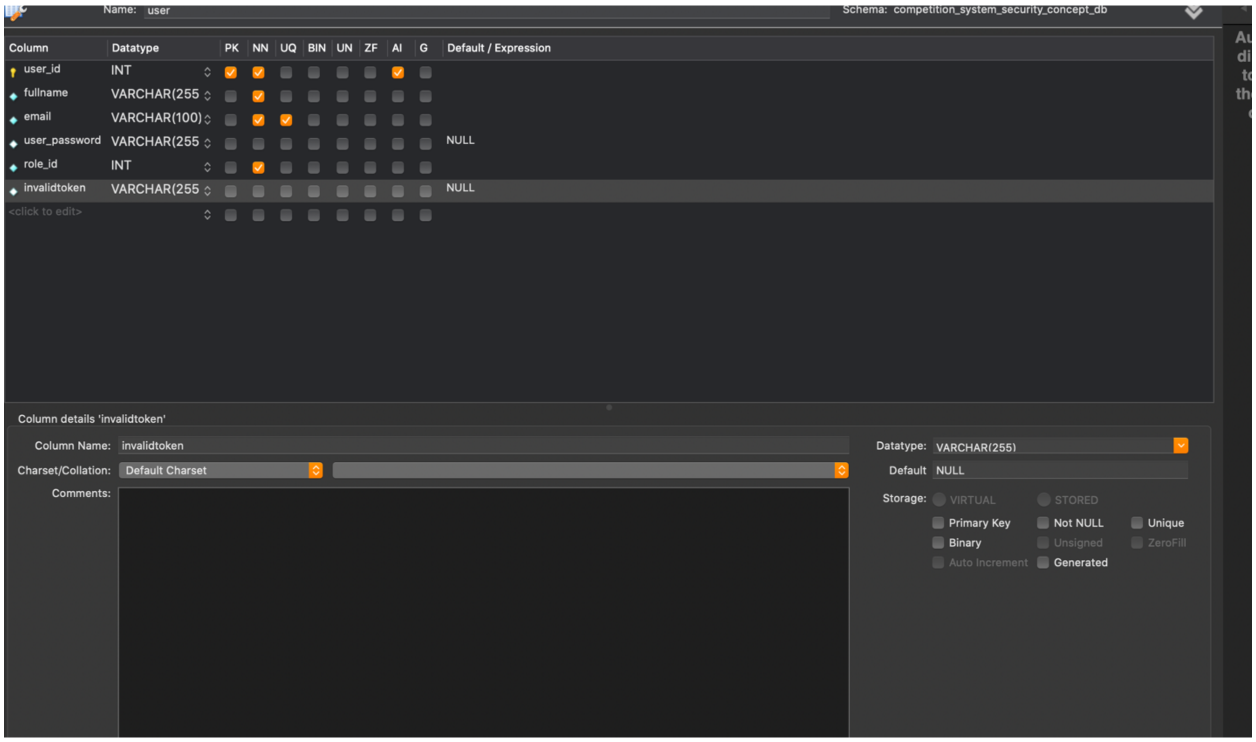
	Value
_name	user
en	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTAxLCJpYXQiOjE2MDkyNTI4NDAsImV4
r_id	101

  - **Close browser tab with [D]esign [C]ompetition and reopen in a new tab and check local storage. Should still see the Rita's userid, token and role.**



Key	Value
role_name	user
token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTAxLCJpYX
user_id	101

- **Recommendations**
    - **TTL**
    - JWT token should be invalidated on server after logout.
- **Solution**
    - **Add a new column called invalidToken in SQL user table.**



Name: user   Schema: competition_system_security_concept_db

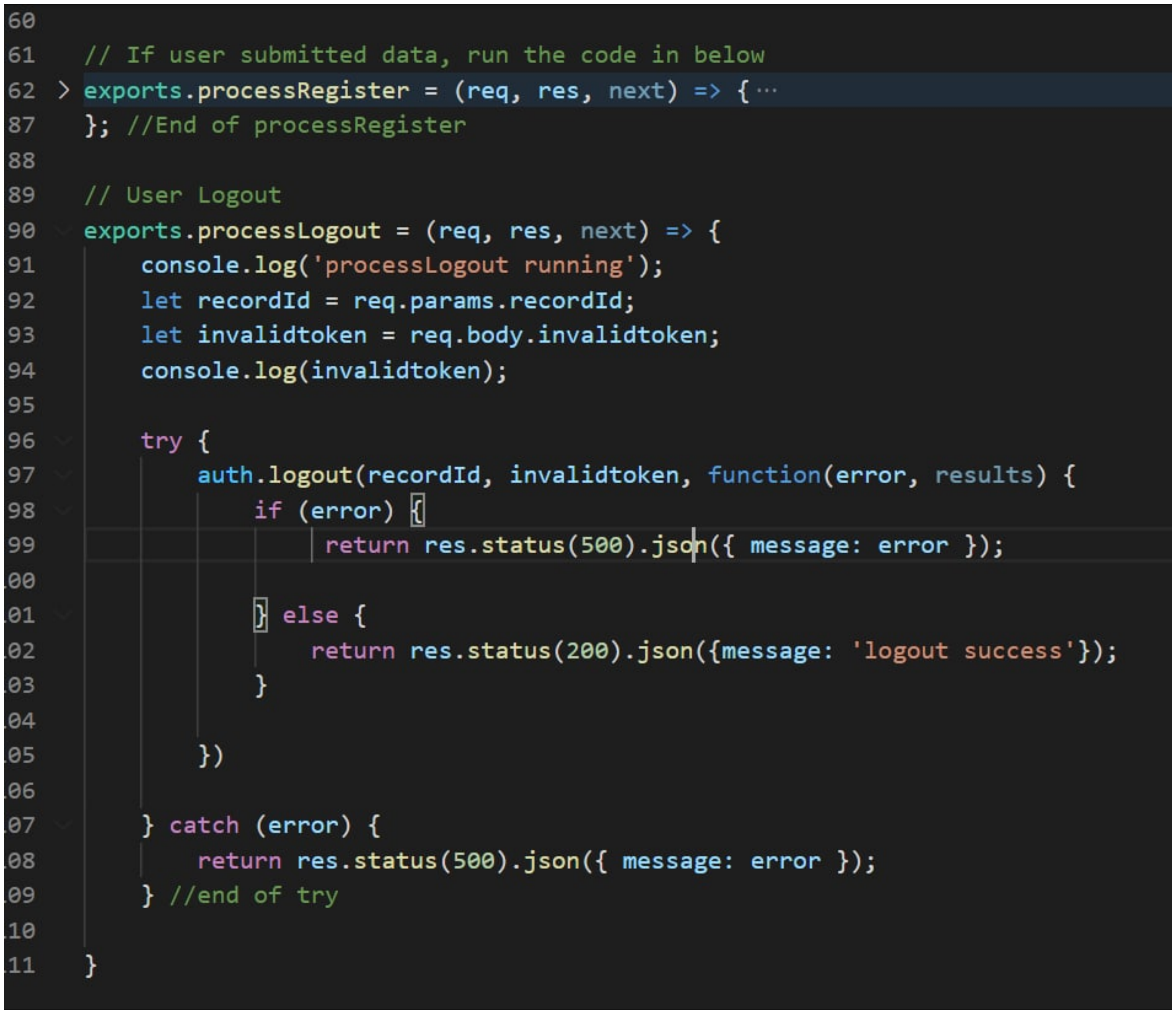
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
user_id	INT	✓	✓					✓		
fullname	VARCHAR(255		✓							
email	VARCHAR(100)		✓	✓						
user_password	VARCHAR(255									NULL
role_id	INT		✓							
invalidtoken	VARCHAR(255									NULL
<click to edit>										

Column details 'invalidtoken'

Column Name: invalidtoken
Charset/Collation: Default Charset
Comments:

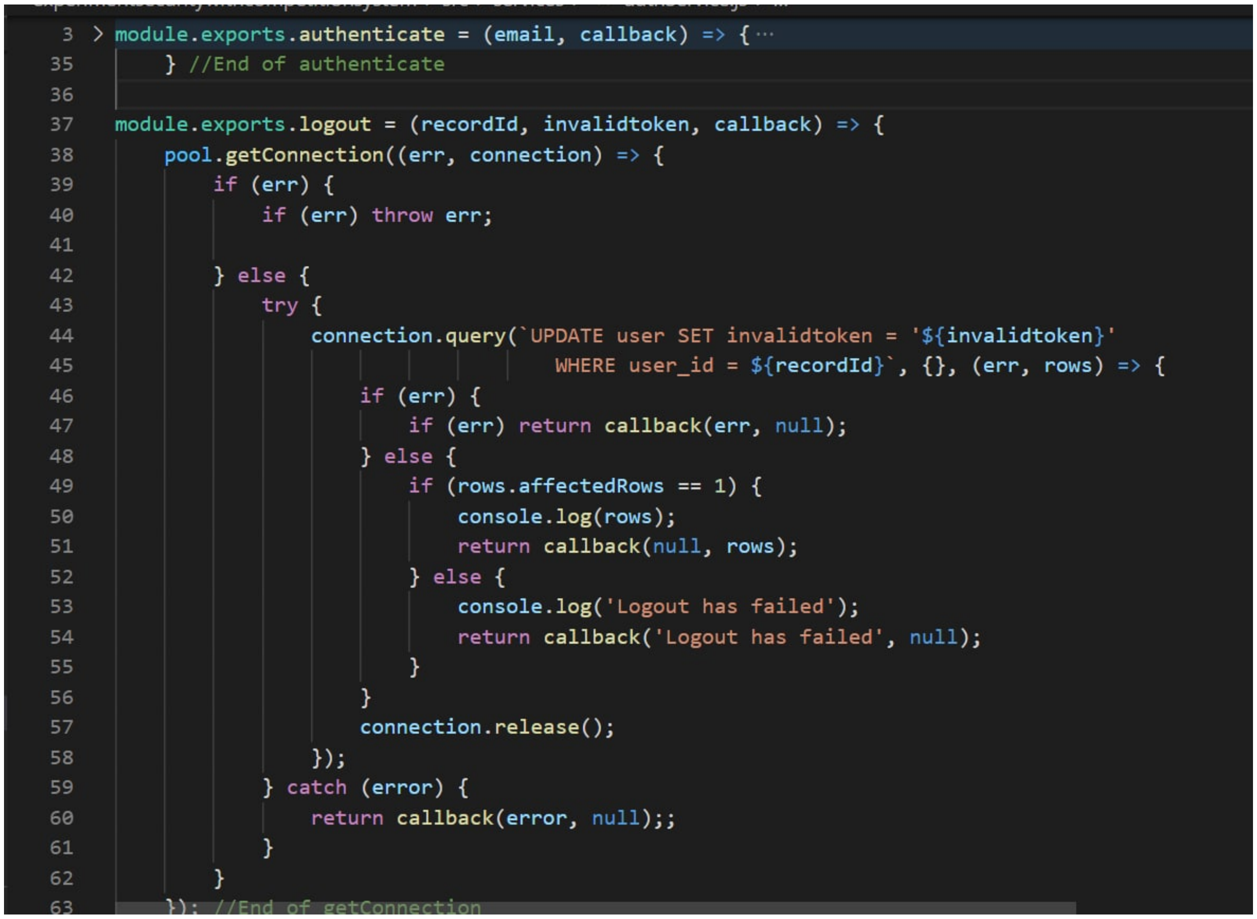
Datatype: VARCHAR(255)
Default NULL
Storage: VIRTUAL STORED
Primary Key Not NULL Unique
Binary Unsigned ZeroFill
Auto Increment Generated

- Create a new process called processLogout in authController.js so that the token will be stored under invalidtoken column in the SQL database.



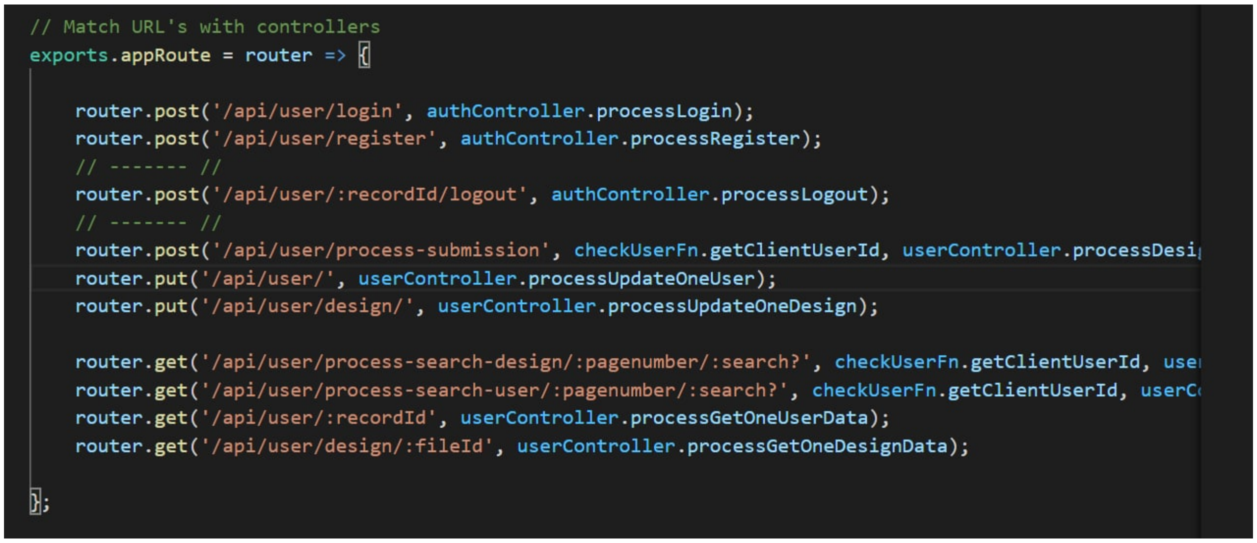
```

// If user submitted data, run the code in below
exports.processRegister = (req, res, next) => {⋯
}; //End of processRegister

// User Logout
exports.processLogout = (req, res, next) => {
    console.log('processLogout running');
    let recordId = req.params.recordId;
    let invalidtoken = req.body.invalidtoken;
    console.log(invalidtoken);

    try {
        auth.logout(recordId, invalidtoken, function(error, results) {
            if (error) {
                return res.status(500).json({ message: error });

            } else {
                return res.status(200).json({message: 'logout success'});
            }

        })

    } catch (error) {
        return res.status(500).json({ message: error });
    } //end of try

}
```

- **As well as in authService.js.**



```
module.exports.authenticate = (email, callback) => {…
    } //End of authenticate

module.exports.logout = (recordId, invalidtoken, callback) => {
    pool.getConnection((err, connection) => {
        if (err) {
            if (err) throw err;

        } else {
            try {
                connection.query(`UPDATE user SET invalidtoken = '${invalidtoken}'
                                  WHERE user_id = ${recordId}`, {}, (err, rows) => {
                    if (err) {
                        if (err) return callback(err, null);
                    } else {
                        if (rows.affectedRows == 1) {
                            console.log(rows);
                            return callback(null, rows);
                        } else {
                            console.log('Logout has failed');
                            return callback('Logout has failed', null);
                        }
                    }
                    connection.release();
                });
            } catch (error) {
                return callback(error, null);;
            }
        }
    }); //End of getConnection
```

- **After adding the codes in authService and authController, add to the routes.js.**

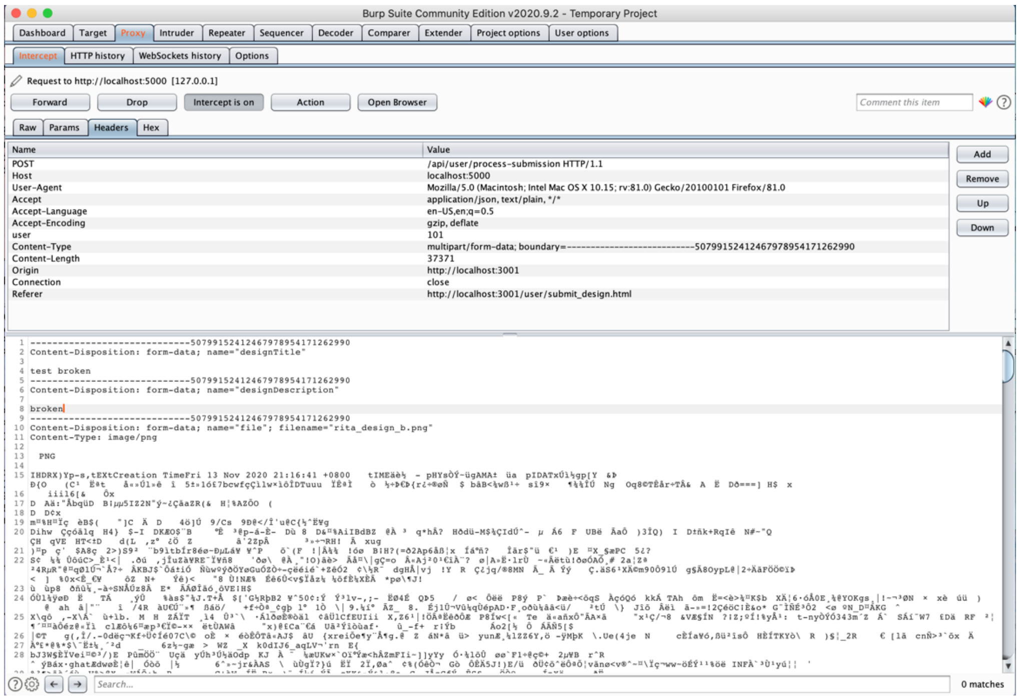


```
// Match URL's with controllers
exports.appRoute = router => {

    router.post('/api/user/login', authController.processLogin);
    router.post('/api/user/register', authController.processRegister);
    // ------- //
    router.post('/api/user/:recordId/logout', authController.processLogout);
    // ------- //
    router.post('/api/user/process-submission', checkUserFn.getClientUserId, userController.processDesi
    router.put('/api/user/', userController.processUpdateOneUser);
    router.put('/api/user/design/', userController.processUpdateOneDesign);

    router.get('/api/user/process-search-design/:pagenumber/:search?', checkUserFn.getClientUserId, use
    router.get('/api/user/process-search-user/:pagenumber/:search?', checkUserFn.getClientUserId, userC
    router.get('/api/user/:recordId', userController.processGetOneUserData);
    router.get('/api/user/design/:fileId', userController.processGetOneDesignData);

};
```

Broken Access Control

Attackers acting as users / admin to create, access, updating or deleting records.

- **Vulnerability level**: High
- **Explanation**: It's a broken access control as the attacker is able to act as another user and are then able to do things that users did not intend to do such as submitting a design. Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification or destruction of all data or performing a business function outside of the limits of user.
- **Detailed Example**
    - **Intercept the submission of the design.**

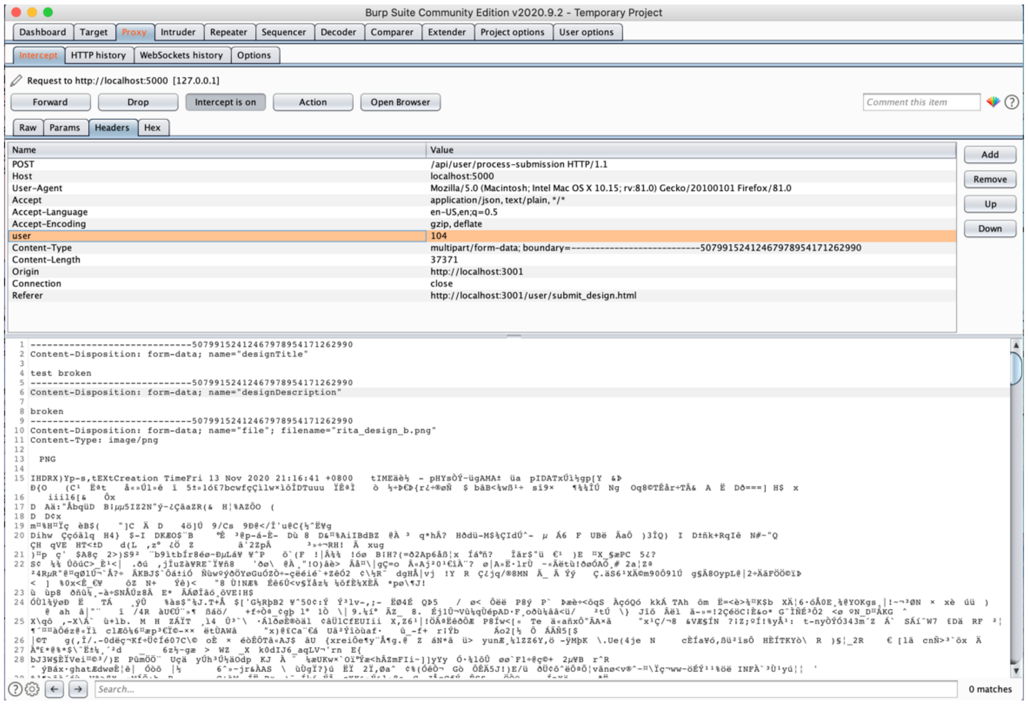


Request to http://localhost:5000 [127.0.0.1]

Name	Value
POST	/api/user/process-submission HTTP/1.1
Host	localhost:5000
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:81.0) Gecko/20100101 Firefox/81.0
Accept	application/json, text/plain, */*
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
user	101
Content-Type	multipart/form-data; boundary=---------------------------50799152412467978954171262990
Content-Length	37371
Origin	http://localhost:3001
Connection	close
Referer	http://localhost:3001/user/submit_design.html

```
-----------------------------50799152412467978954171262990
Content-Disposition: form-data; name="designTitle"

test broken
-----------------------------50799152412467978954171262990
Content-Disposition: form-data; name="designDescription"

broken
-----------------------------50799152412467978954171262990
Content-Disposition: form-data; name="file"; filename="rita_design_b.png"
Content-Type: image/png

 PNG

IHDRX)Yp-s,tEXtCreation TimeFri 13 Nov 2020 21:16:41 +0800 [illegible]
```

- **Change User params to any user Id (User Id : 104) and forward request.**

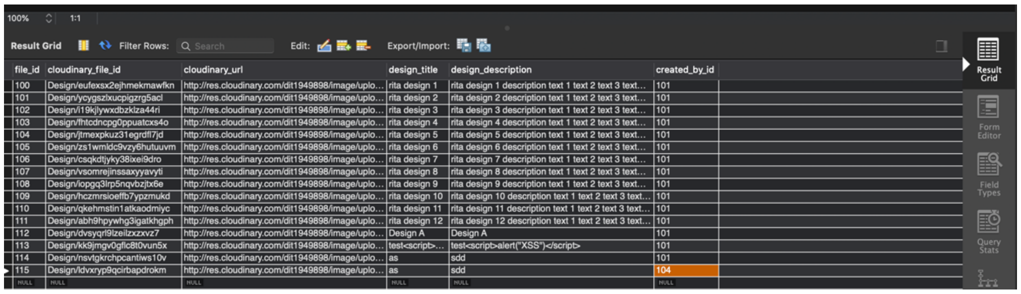


Request to http://localhost:5000 [127.0.0.1]

Name	Value
POST	/api/user/process-submission HTTP/1.1
Host	localhost:5000
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:81.0) Gecko/20100101 Firefox/81.0
Accept	application/json, text/plain, */*
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
user	104
Content-Type	multipart/form-data; boundary=---------------------------50799152412467978954171262990
Content-Length	37371
Origin	http://localhost:3001
Connection	close
Referer	http://localhost:3001/user/submit_design.html

```
-----------------------------50799152412467978954171262990
Content-Disposition: form-data; name="designTitle"

test broken
-----------------------------50799152412467978954171262990
Content-Disposition: form-data; name="designDescription"

broken
-----------------------------50799152412467978954171262990
Content-Disposition: form-data; name="file"; filename="rita_design_b.png"
Content-Type: image/png

 PNG

IHDRX)Yp-s,tEXtCreation TimeFri 13 Nov 2020 21:16:41 +0800   tIMEäè½ - pHYsÒÝ-ügAMA± üa pIDATxÚì½gp[Y &Þ
[illegible]
```

- **Design Submitted as User Id: 104**



file_id	cloudinary_file_id	cloudinary_url	design_title	design_description	created_by_id
100	Design/eufexsx2ejhmekmawfkn	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 1	rita design 1 description text 1 text 2 text 3 text...	101
101	Design/ycygszlxucpigzrg5acl	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 2	rita design 2 description text 1 text 2 text 3 text...	101
102	Design/i19kjlywxdbzklza44ri	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 3	rita design 3 description text 1 text 2 text 3 text...	101
103	Design/fhtcdncpg0ppuatcxs4o	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 4	rita design 4 description text 1 text 2 text 3 text...	101
104	Design/jtmexpkuz31egrdfl7jd	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 5	rita design 5 description text 1 text 2 text 3 text...	101
105	Design/zs1wmldc9vzy6hutuuvm	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 6	rita design 6 description text 1 text 2 text 3 text...	101
106	Design/csqkdtjyky38ixei9dro	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 7	rita design 7 description text 1 text 2 text 3 text...	101
107	Design/vsomrejinssaxyyavyti	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 8	rita design 8 description text 1 text 2 text 3 text...	101
108	Design/iopgq3lrp5nqvbzjtx6e	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 9	rita design 9 description text 1 text 2 text 3 text...	101
109	Design/hczmrsioeffb7ypzmukd	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 10	rita design 10 description text 1 text 2 text 3 text...	101
110	Design/qkehmstin1atkaodmiyc	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 11	rita design 11 description text 1 text 2 text 3 text...	101
111	Design/abh9hpywhg3igatkhgph	http://res.cloudinary.com/dit1949898/image/uplo...	rita design 12	rita design 12 description text 1 text 2 text 3 text...	101
112	Design/dvsyqrl9lzeilzxzxvz7	http://res.cloudinary.com/dit1949898/image/uplo...	Design A	Design A	101
113	Design/kk9jmgv0gflc8t0vun5x	http://res.cloudinary.com/dit1949898/image/uplo...	test<script>...	test<script>alert("XSS")</script>	101
114	Design/nsvtgkrchpcantiws10v	http://res.cloudinary.com/dit1949898/image/uplo...	as	sdd	101
115	Design/ldvxryp9qcirbapdrokm	http://res.cloudinary.com/dit1949898/image/uplo...	as	sdd	104
NULL	NULL	NULL	NULL	NULL	NULL

- **Recommendations**
  - To prevent broken access control, there should be a trusted server code or serverless API where attacker cannot modify the access control check where the attacker cannot modify the access control check or metadata.
- **Solution**
  - **Same as broken authentication solution.**

Sensitive Data Exposure

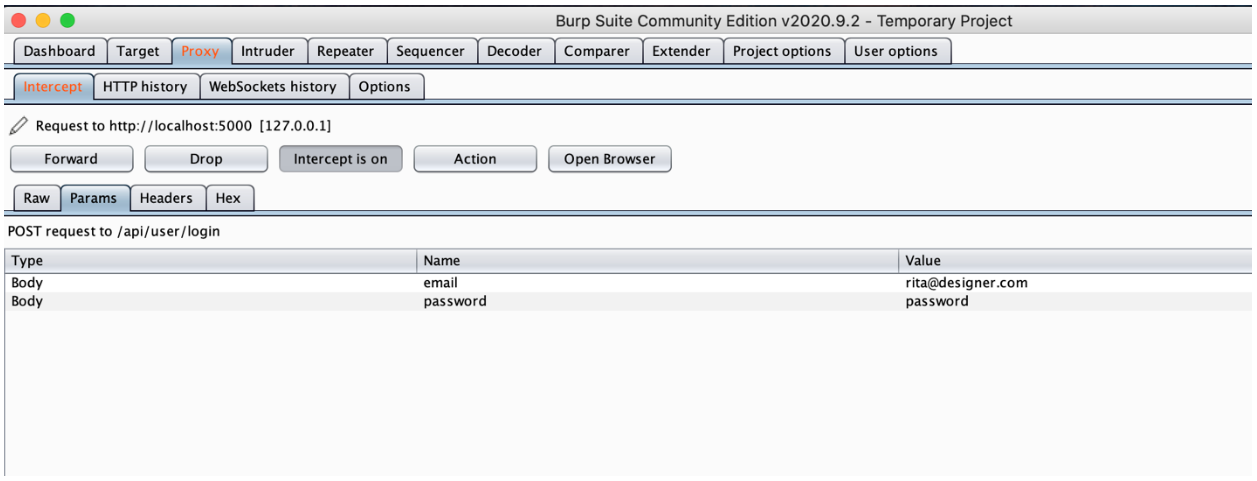
Failure frequently compromises all data that should have been protected.

- **Vulnerability level**: High
- **Explanation**: The most common flaw and impactful attack. Rather than directly attacking crypto, attackers steak keys, execute man in the middle attacks or steal clear text data off the server while in transit or from the user's client e.g., browsers. The password database uses unsalted or simple hashes to store passwords hence attacker

can upload a file to retrieve password from database. All the unsalted hashes can be exposed with a table of already pre-calculated hashes.

  - **Hashes generated by simple or fast hash functions maybe cracked by GPUs (graphic processor), even they were salted.**

- **Detailed Example**
  - **Intercept login for Rita and able to see sensitive data like her password.**



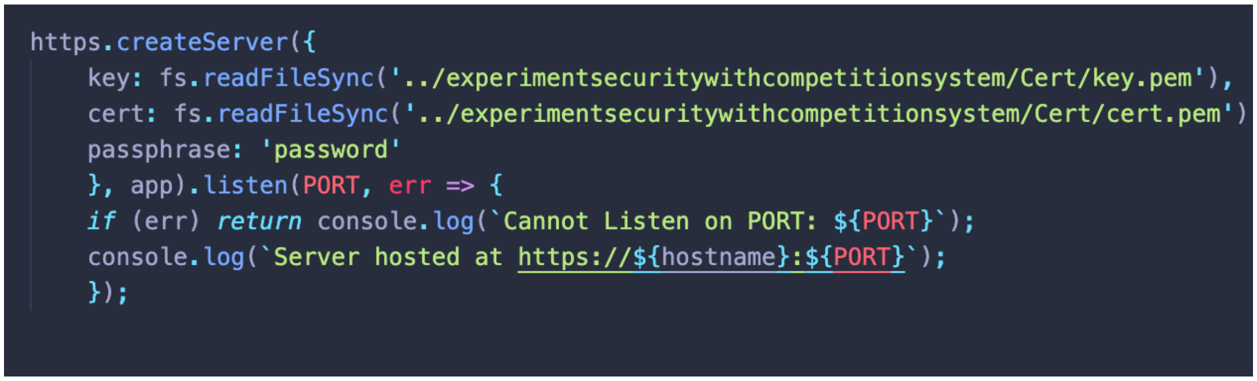
Request to http://localhost:5000 [127.0.0.1]

POST request to /api/user/login

Type	Name	Value
Body	email	rita@designer.com
Body	password	password

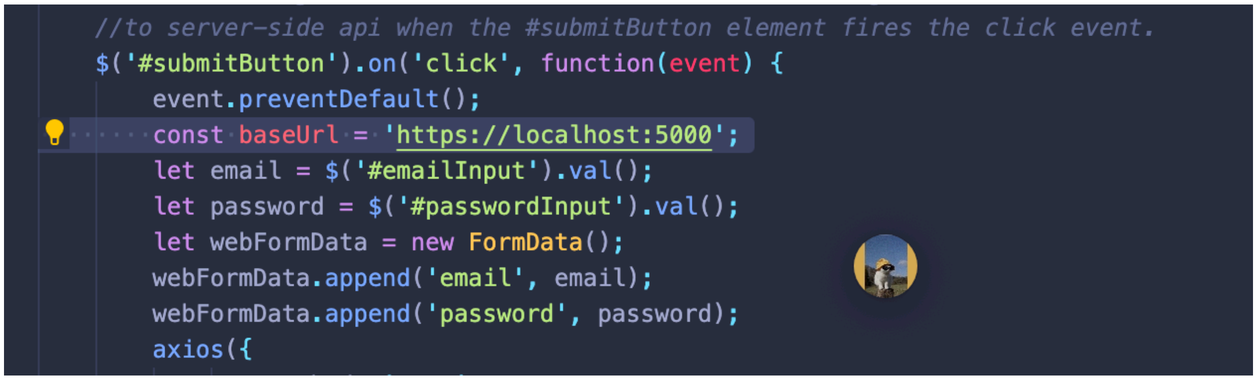
- **Recommendation**
  - **SSL Certificate**
    - **SSL stands for Secure Socket Layer, it's the standard technology for keeping an internet connection secure and safeguarding any sensitive data that is being sent between two systems, preventing criminals from reading and modifying any information transferred, including potential personal details**
  - **With the help of an SSL certificate, enable websites to move from HTTP to HTTPS, which is more secure. An SSL certificate is a data file hosted in a website's origin server. SSL certificates make SSL/TLS encryption possible, and they contain the website's public key and the website's identity, along with related information.**
- **Solution**
  - **Use the command in backend folder**
    - `openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365`
      - This commands means that it gemetates to create a new root SSl certificate and then saved it to a file name rootCA.pem. This certificate will have a validity of 1024 days, able to change it to any number of days you want. Will also prompt you for additional optional information.
    - You should see a cert.pem and key.perm in your backend folder
    - Create a folder called Cert and put both perm folder in Cer folder.
  - **They will ask for a password so just type in password.**
    - Country Name (2 letter code) []:SG
    - State or Province Name (full name) []:Singapore
    - Locality Name (eg, city) []:Singapore
    - Organization Name (eg, company) []:SP

        - Organizational Unit Name (eg, section) []:SOC
        - Common Name (eg, fully qualified host name) []:Localhost
        - Email Address []:o
    - **Afterwards, install HTTPS via NPM**
        - NPM INSTALL HTTPS
    - **Type in this line of code on index.js for frontend and backend**



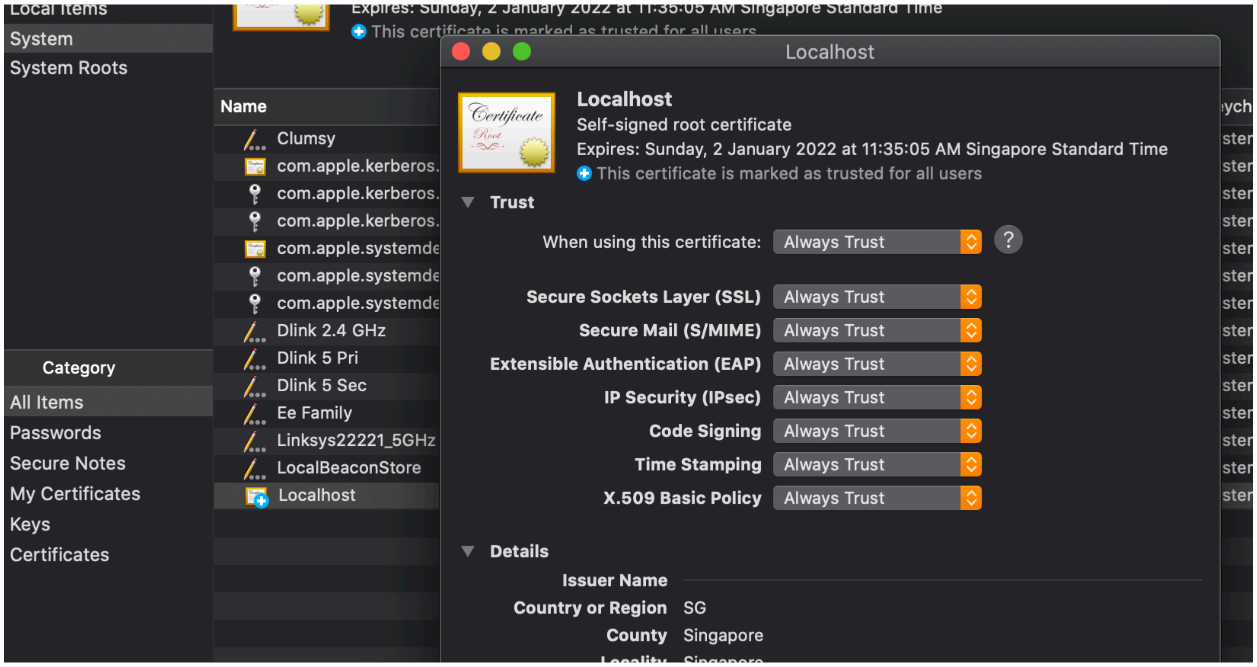
```
https.createServer({
    key: fs.readFileSync('../experimentsecuritywithcompetitionsystem/Cert/key.pem'),
    cert: fs.readFileSync('../experimentsecuritywithcompetitionsystem/Cert/cert.pem')
    passphrase: 'password'
    }, app).listen(PORT, err => {
    if (err) return console.log(`Cannot Listen on PORT: ${PORT}`);
    console.log(`Server hosted at https://${hostname}:${PORT}`);
    });
```

    - **Change all HTTP to HTTPS in all JS file in Frontend. Example:**



```
    //to server-side api when the #submitButton element fires the click event.
    $('#submitButton').on('click', function(event) {
        event.preventDefault();
        const baseUrl = 'https://localhost:5000';
        let email = $('#emailInput').val();
        let password = $('#passwordInput').val();
        let webFormData = new FormData();
        webFormData.append('email', email);
        webFormData.append('password', password);
        axios({
```

    - **Install certificate (Cert.pem) into keychain access for MACBOOK users. And ensure trust is "always trust" under the option.**



Localhost

Self-signed root certificate

Expires: Sunday, 2 January 2022 at 11:35:05 AM Singapore Standard Time

This certificate is marked as trusted for all users

Trust

When using this certificate:	Always Trust
Secure Sockets Layer (SSL)	Always Trust
Secure Mail (S/MIME)	Always Trust
Extensible Authentication (EAP)	Always Trust
IP Security (IPsec)	Always Trust
Code Signing	Always Trust
Time Stamping	Always Trust
X.509 Basic Policy	Always Trust

Details

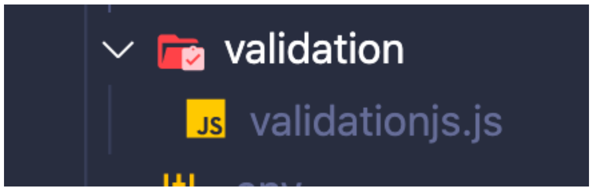
Issuer Name

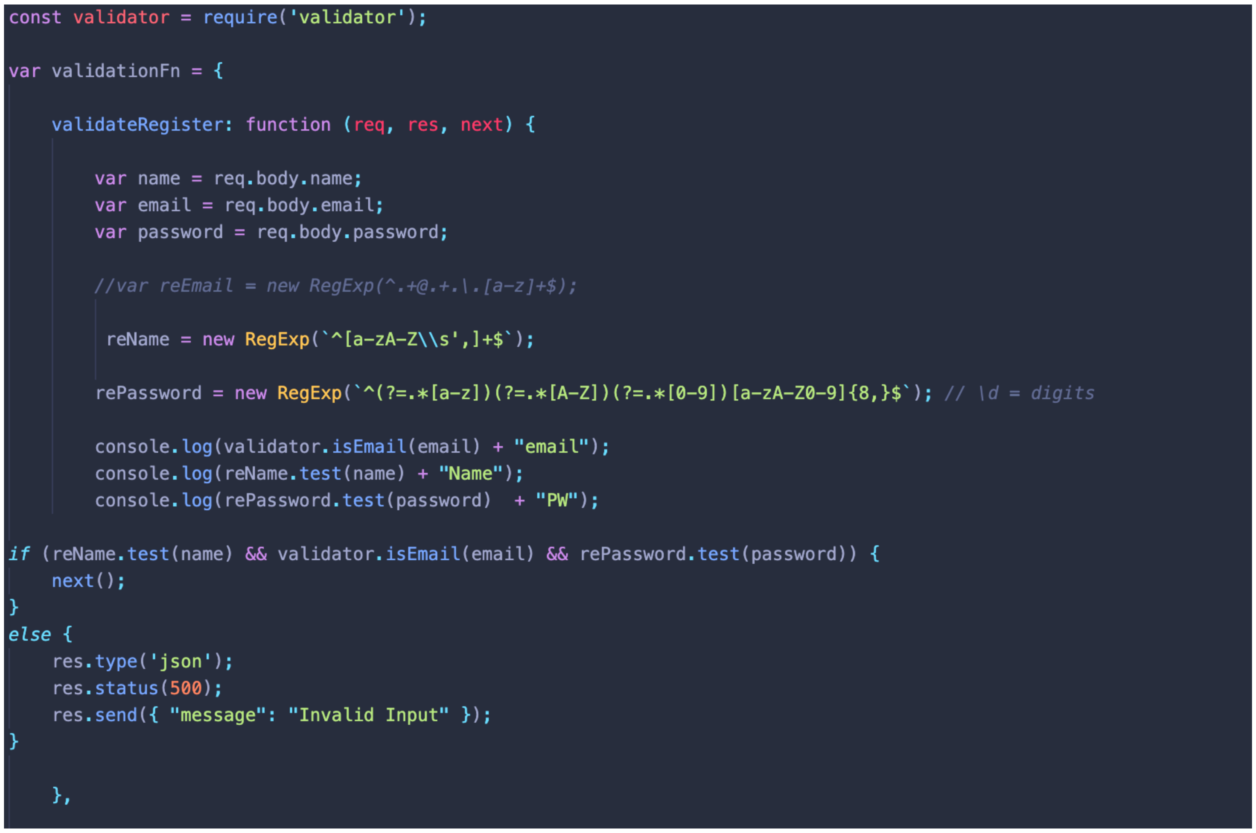
Country or Region	SG
County	Singapore

Cross-site scripting (XSS)

Second most prevalent issue according to OWASP, usually targeting users' browsers.

- **Vulnerability level**: High
- **Explanation**: No validation
- **Recommendation**
  - **Regex**
  - **Validation folder**
- **Solution**
  - **Created a Validation folder, under the folder, created a JavaScript file called validation to validate credentials for existing or registered user.**





```
const validator = require('validator');

var validationFn = {

    validateRegister: function (req, res, next) {

        var name = req.body.name;
        var email = req.body.email;
        var password = req.body.password;

        //var reEmail = new RegExp(^.+@.+.\.[a-z]+$);

         reName = new RegExp(`^[a-zA-Z\\s',]+$`);

        rePassword = new RegExp(`^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-9]{8,}$`); // \d = digits

        console.log(validator.isEmail(email) + "email");
        console.log(reName.test(name) + "Name");
        console.log(rePassword.test(password)  + "PW");

if (reName.test(name) && validator.isEmail(email) && rePassword.test(password)) {
    next();
}
else {
    res.type('json');
    res.status(500);
    res.send({ "message": "Invalid Input" });
}

    },
```

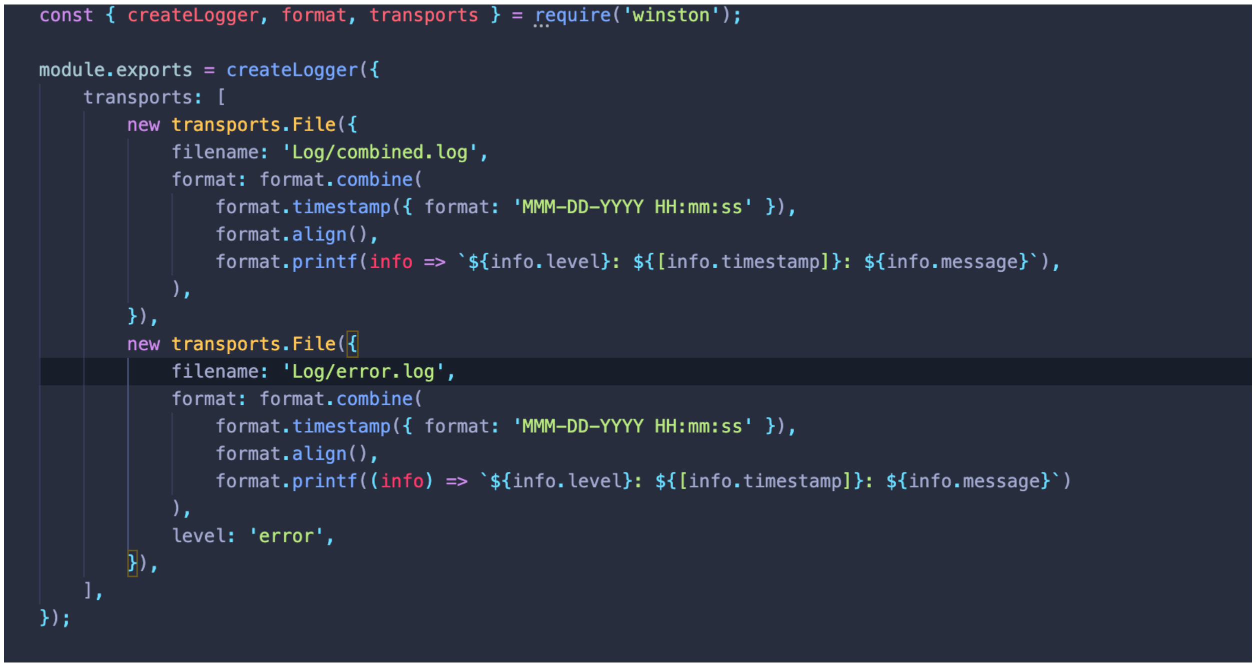
- **Install NPM (node package manager) package called validator via the command prompt or terminal (NPM install validator). Validator package is a library of string validators and sanitizers.**
- **Created a function called ValidateFn and under the function, there is a validateRegister Function where it validates the name, email and password of the user who is registering for an account. It validates the data sent by checking the regex, known as Regular expression, a sequence of characters that define a search pattern.**
    - **For the variable of name Regex, it checks if the input data contains letters only**
    - **For email, we will be using the isEmail validator in built function, it checks if the string from user input is an valid email.**
    - **For password, it checks for whether user input password is minimum 8 characters long, with at least a number and capital letter.**

Insufficient Logging & Monitoring

Vulnerable to information leakage if making logging and alerting events visible to an attacker.

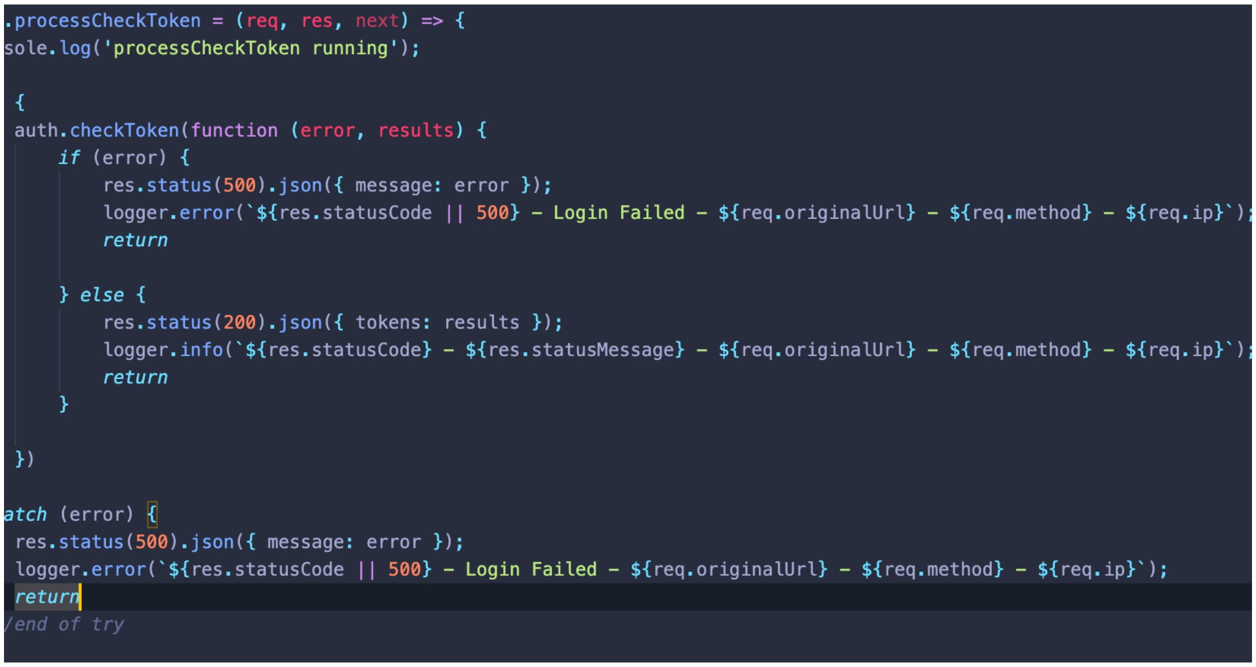
- **Vulnerability level**: High
- **Explanation**: exploitation of insufficient logging and monitoring is when attackers rely on the lack of monitoring and timely response to achieve their goals without being detected, for example, failed logins, no warning and errors generated or unclear log messages,

- **Recommendation**
  - **Ensure all login, access control failures and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts.**
  - **Ensure that logs are generated in a format that can be easily understandable.**
  - **Establish an incident response and recovery plan.**
- **Solution**
  - **Create a folder called Log under backend folder**
    - **Create 2 files called error.log and combined.log**
  - **NPM INSTALL WINSTON**
  - Create a new file under services called loggingService.js
    - Add these codes under the file



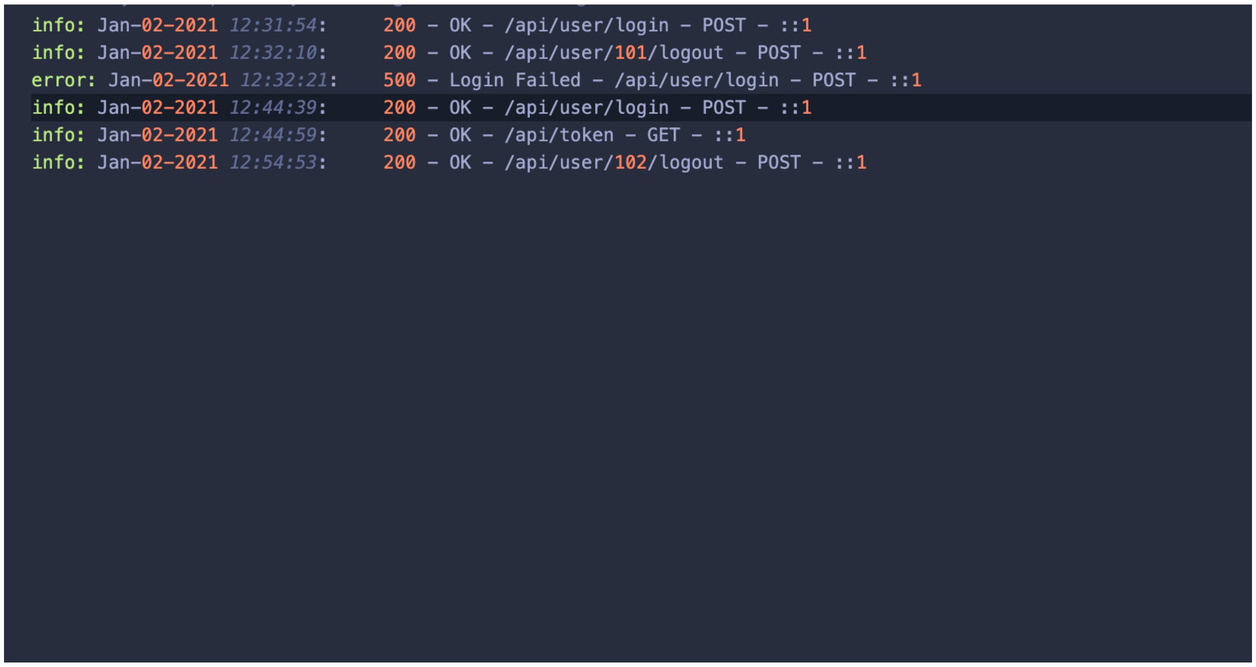
```
const { createLogger, format, transports } = require('winston');

module.exports = createLogger({
    transports: [
        new transports.File({
            filename: 'Log/combined.log',
            format: format.combine(
                format.timestamp({ format: 'MMM-DD-YYYY HH:mm:ss' }),
                format.align(),
                format.printf(info => `${info.level}: ${[info.timestamp]}: ${info.message}`),
            ),
        }),
        new transports.File({
            filename: 'Log/error.log',
            format: format.combine(
                format.timestamp({ format: 'MMM-DD-YYYY HH:mm:ss' }),
                format.align(),
                format.printf((info) => `${info.level}: ${[info.timestamp]}: ${info.message}`)
            ),
            level: 'error',
        }),
    ],
});
```

  - Change the return messages in authController and userController.js
    - For 500 and 200 code messages



```
.processCheckToken = (req, res, next) => {
sole.log('processCheckToken running');

 {
 auth.checkToken(function (error, results) {
     if (error) {
         res.status(500).json({ message: error });
         logger.error(`${res.statusCode || 500} - Login Failed - ${req.originalUrl} - ${req.method} - ${req.ip}`);
         return

     } else {
         res.status(200).json({ tokens: results });
         logger.info(`${res.statusCode} - ${res.statusMessage} - ${req.originalUrl} - ${req.method} - ${req.ip}`);
         return
     }

 })

atch (error) {
 res.status(500).json({ message: error });
 logger.error(`${res.statusCode || 500} - Login Failed - ${req.originalUrl} - ${req.method} - ${req.ip}`);
 return
/end of try
```

- Try logging in and check combineLog.js for any existing log



```
  info: Jan-02-2021 12:31:54:     200 - OK - /api/user/login - POST - ::1
  info: Jan-02-2021 12:32:10:     200 - OK - /api/user/101/logout - POST - ::1
  error: Jan-02-2021 12:32:21:    500 - Login Failed - /api/user/login - POST - ::1
  info: Jan-02-2021 12:44:39:     200 - OK - /api/user/login - POST - ::1
  info: Jan-02-2021 12:44:59:     200 - OK - /api/token - GET - ::1
  info: Jan-02-2021 12:54:53:     200 - OK - /api/user/102/logout - POST - ::1
```

Tools & Method for Testing

Include major points and identify the opportunity. Restate the client's project goals you identified previously (such as via RFP, interview, etc.).

- Goal #1: Train all CSRs on new system within 6 weeks of go-live date

- Goal #2: Integrate sales training with functional training on new system
- Goal #3: Monitor sales volume, return rates, and key satisfaction metrics for 6 weeks following training

CONCLUSION

In conclusion, we have documented all the possible vulnerabilities that can occur on the application as well as the solutions to counter these vulnerabilities.