SECURE SOFTWARE 2025

PROJECT PROPOSAL

Presented for :

ENG. Mohamed Hatem

Presented by :

Mohammad Wael 2305035

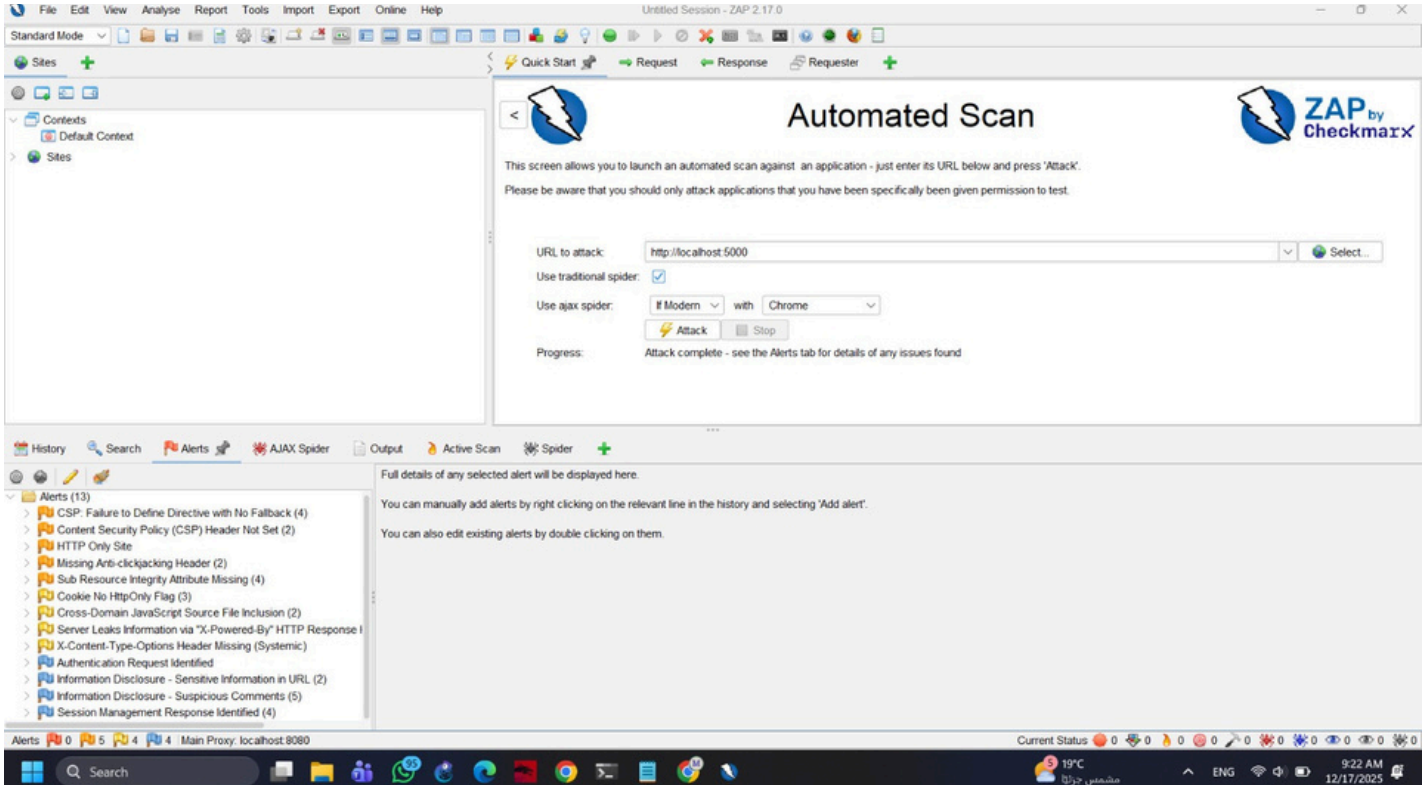
Mahmoud Wael 2305036

Zainab El sayed 2305112

Section 1 (Dast finding) :

A1 : Automated scan :

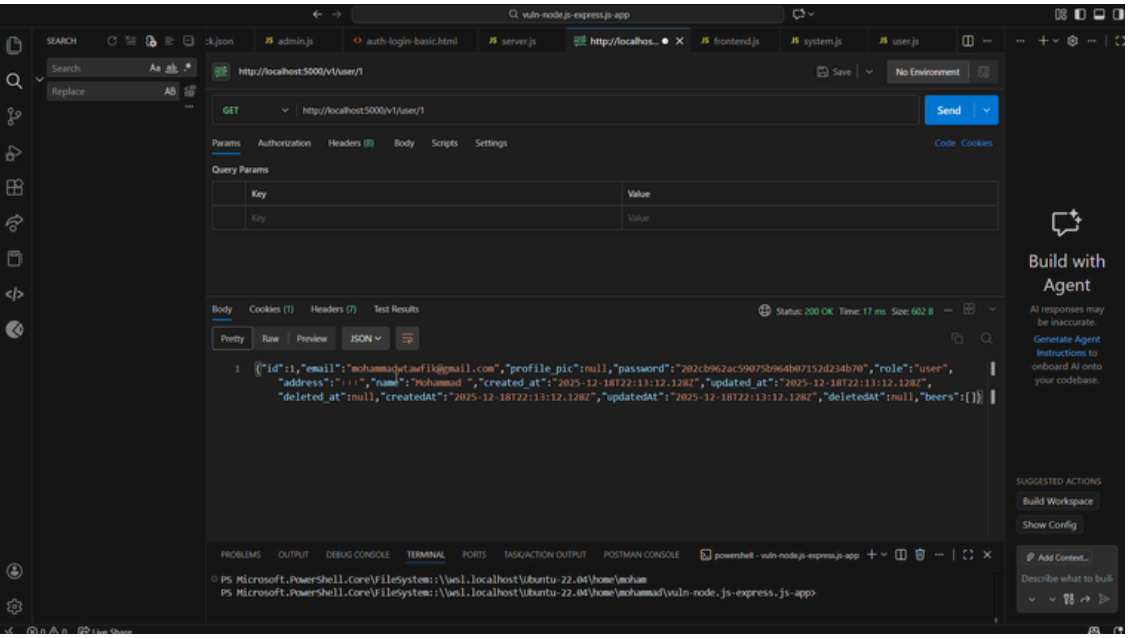
-owasp zap output :



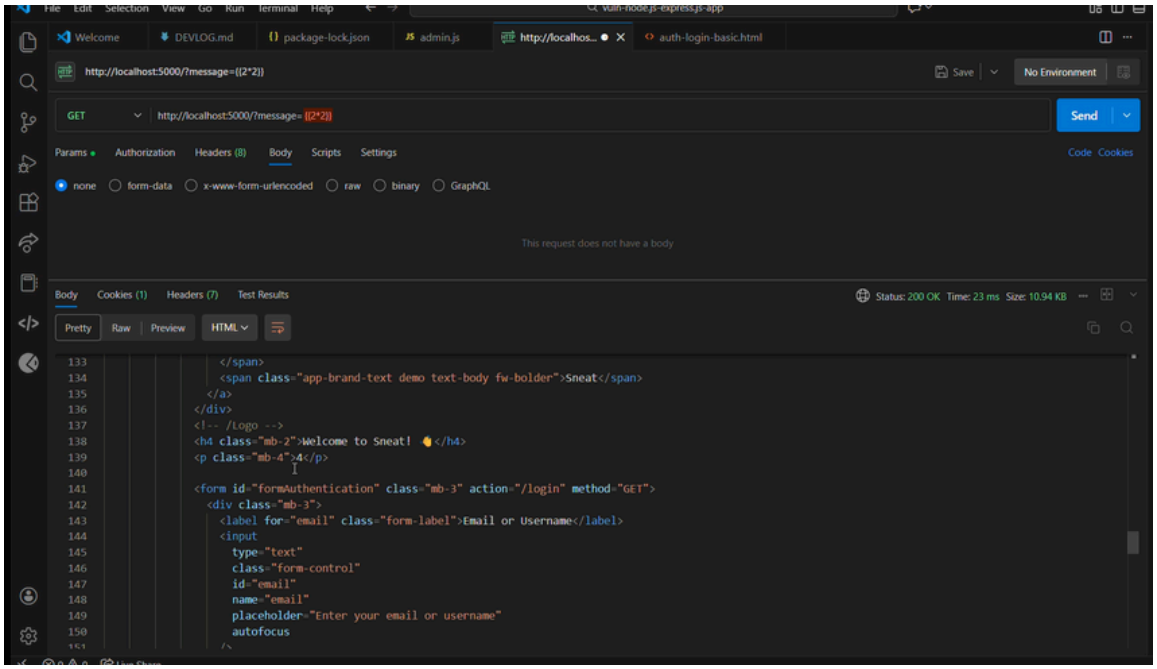
A2 : Manual testing :

-postman output :

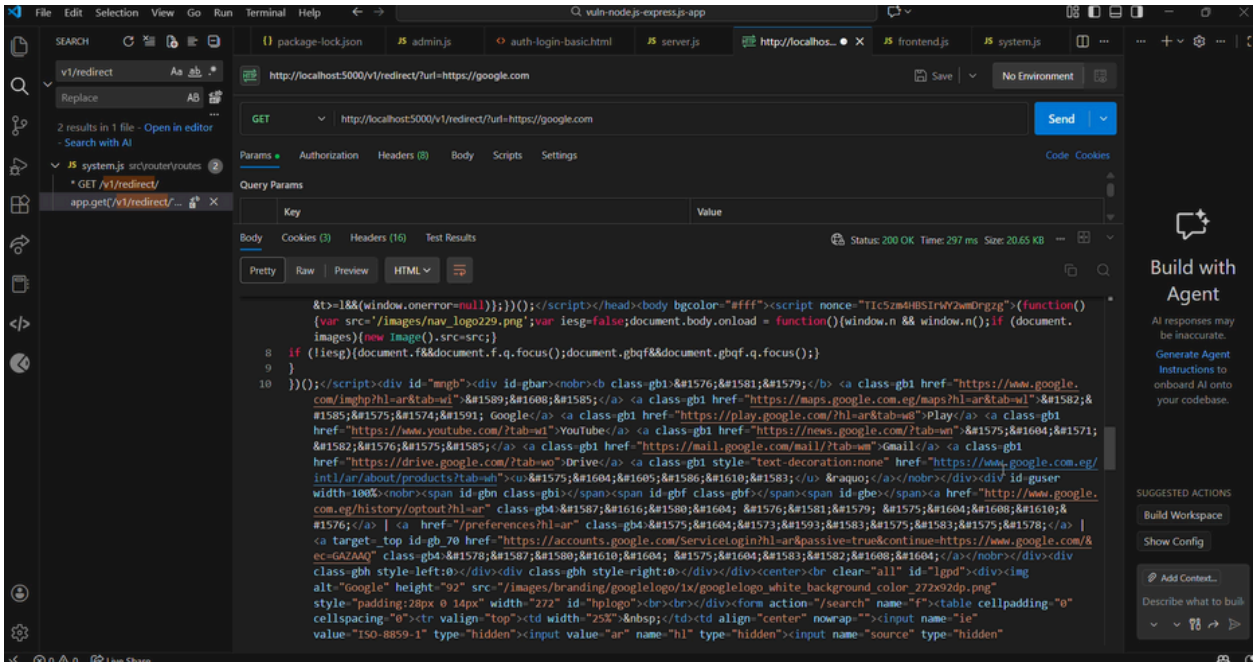
-Broken object level Authorization(unauthorize access):



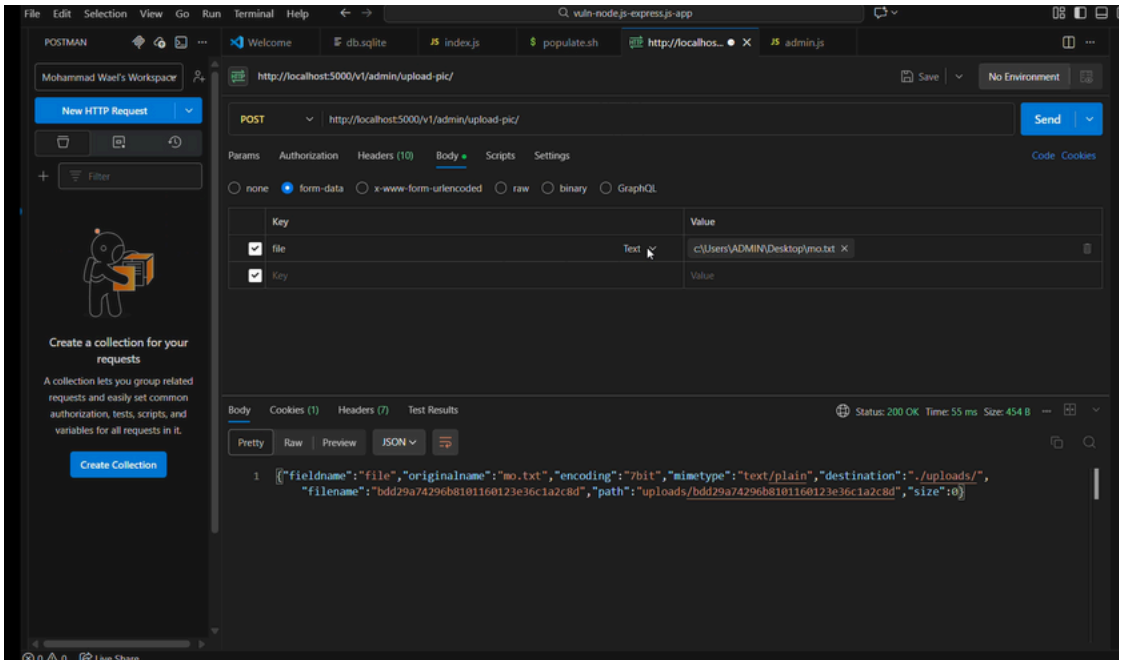
-Reflected xss:



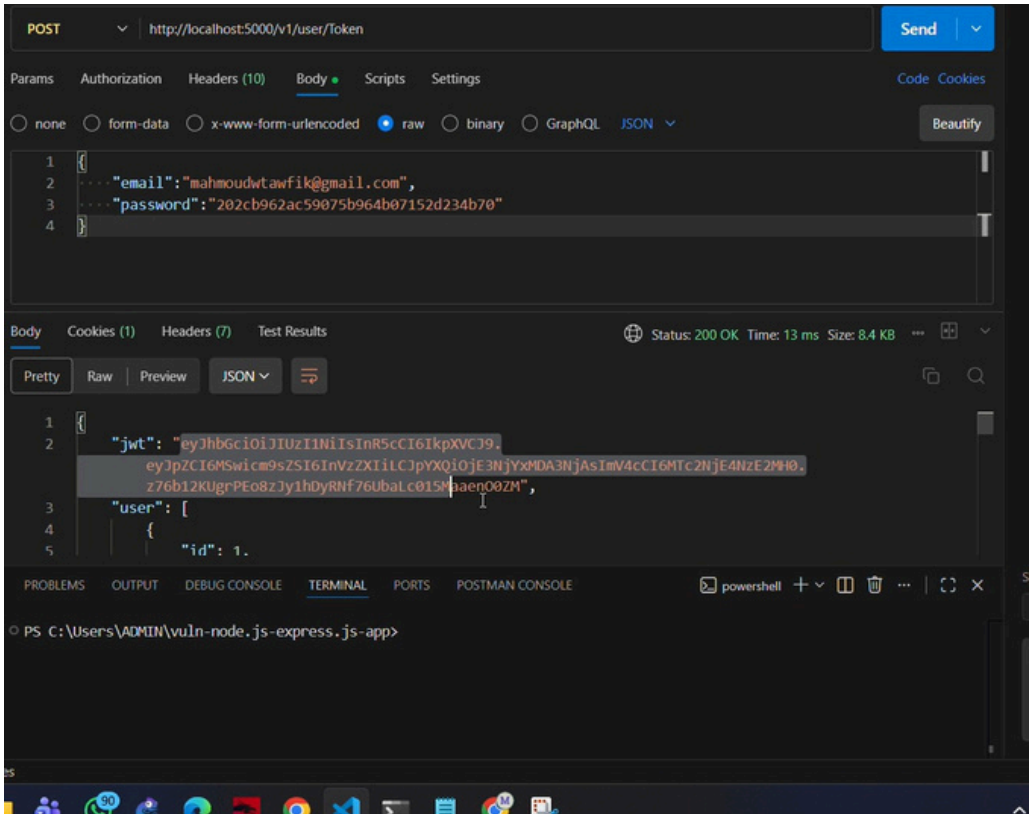
-Open redirect:

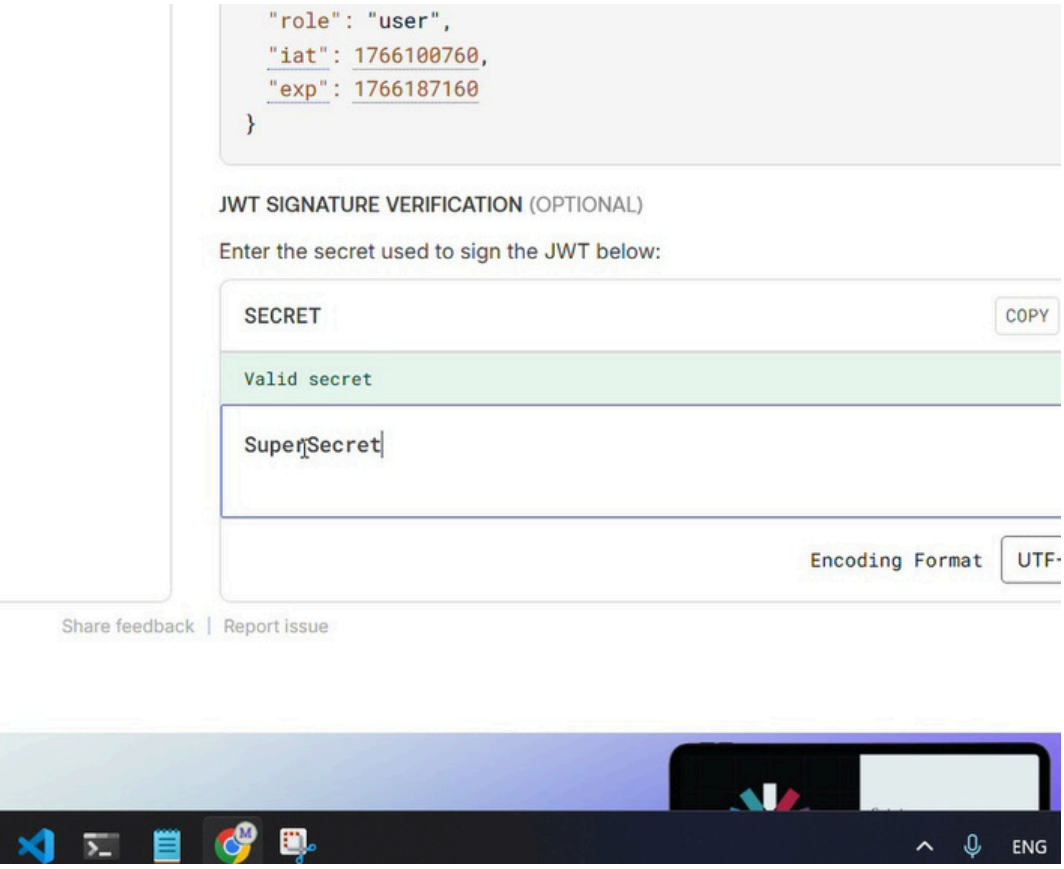


-Input Invalidation (Unrestricted File Upload):

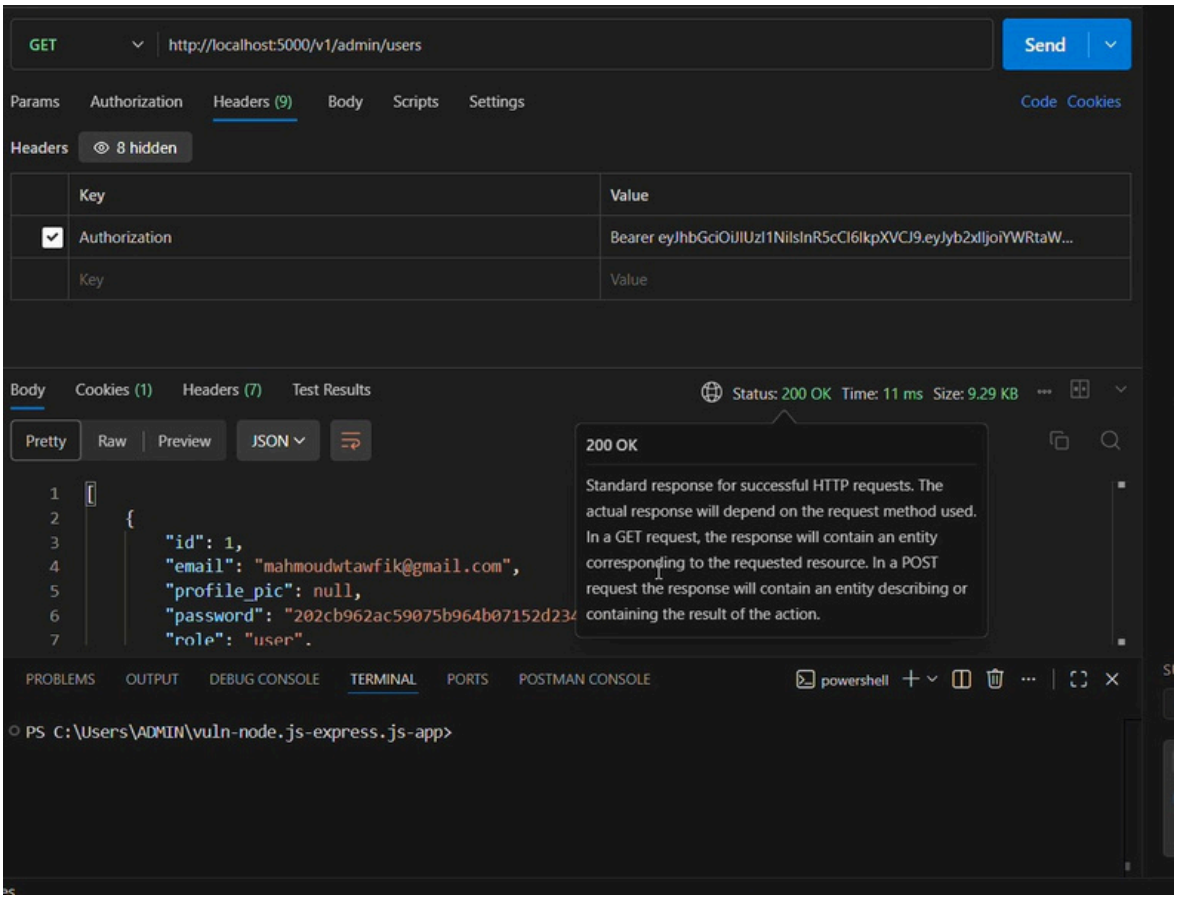


-Weak secret:

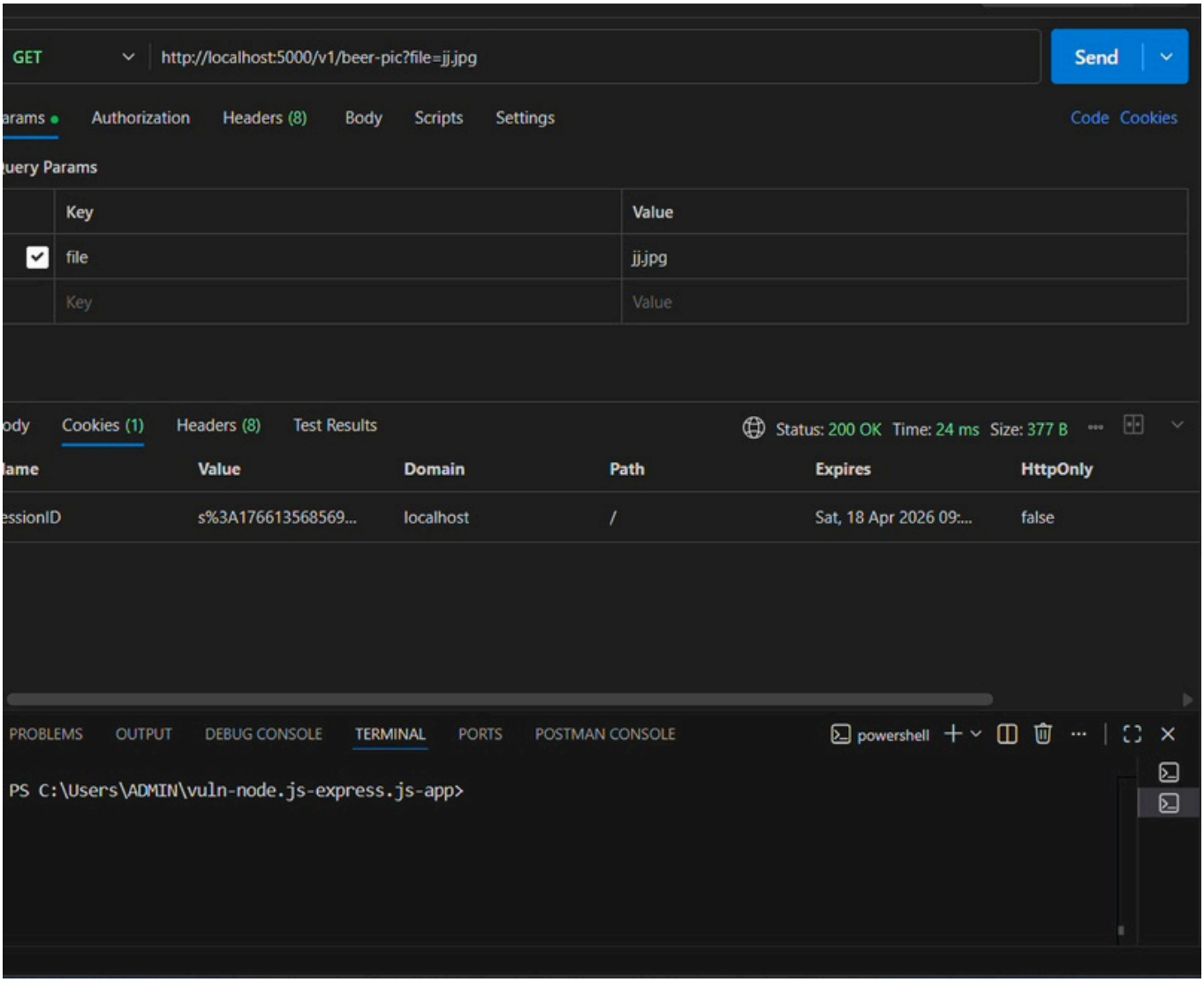

POST
http://localhost:5000/v1/user/Token
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Code
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"email":"mahmoudwtawfik@gmail.com",
"password":"202cb962ac59075b964b07152d234b70"
}
Body
Cookies (1)
Headers (7)
Test Results
Status: 200 OK
Time: 13 ms
Size: 8.4 KB
Pretty
Raw
Preview
JSON
{
"jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6MSwicm9sZSI6InVzZXIiLCJpYXQiOjE3NjYxMDA3NjAsImV4cCI6MTc2NjE4NzE2MH0.
z76b12KUgrPEo8zJy1hDyRNf76UbaLc015MaaenO0ZM",
"user": [
{
"id": 1,
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
powershell
PS C:\Users\ADMIN\vuln-node.js-express.js-app>


"role": "user",
"iat": 1766100760,
"exp": 1766187160
}
JWT SIGNATURE VERIFICATION (OPTIONAL)
Enter the secret used to sign the JWT below:
SECRET
COPY
Valid secret
SuperSecret
Encoding Format
UTF-
Share feedback | Report issue
ENG

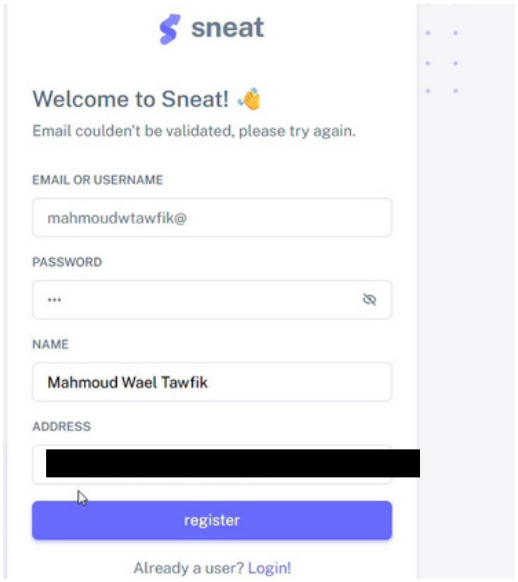
-Unverify jwt:



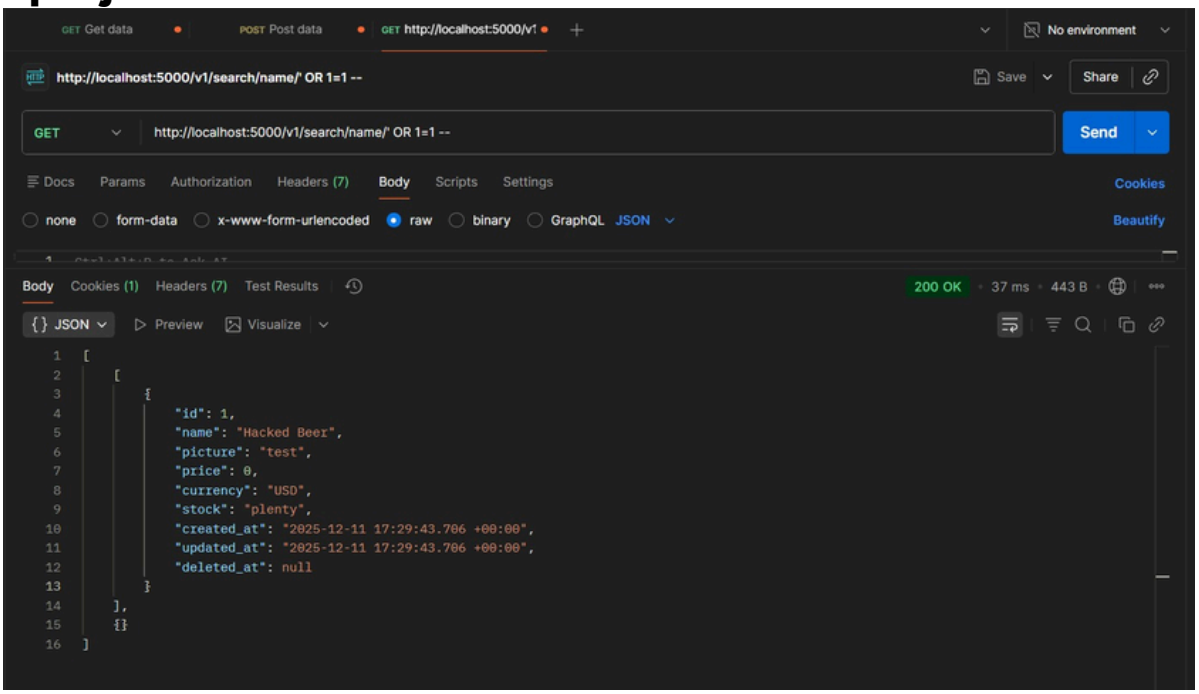
-Traversal path:



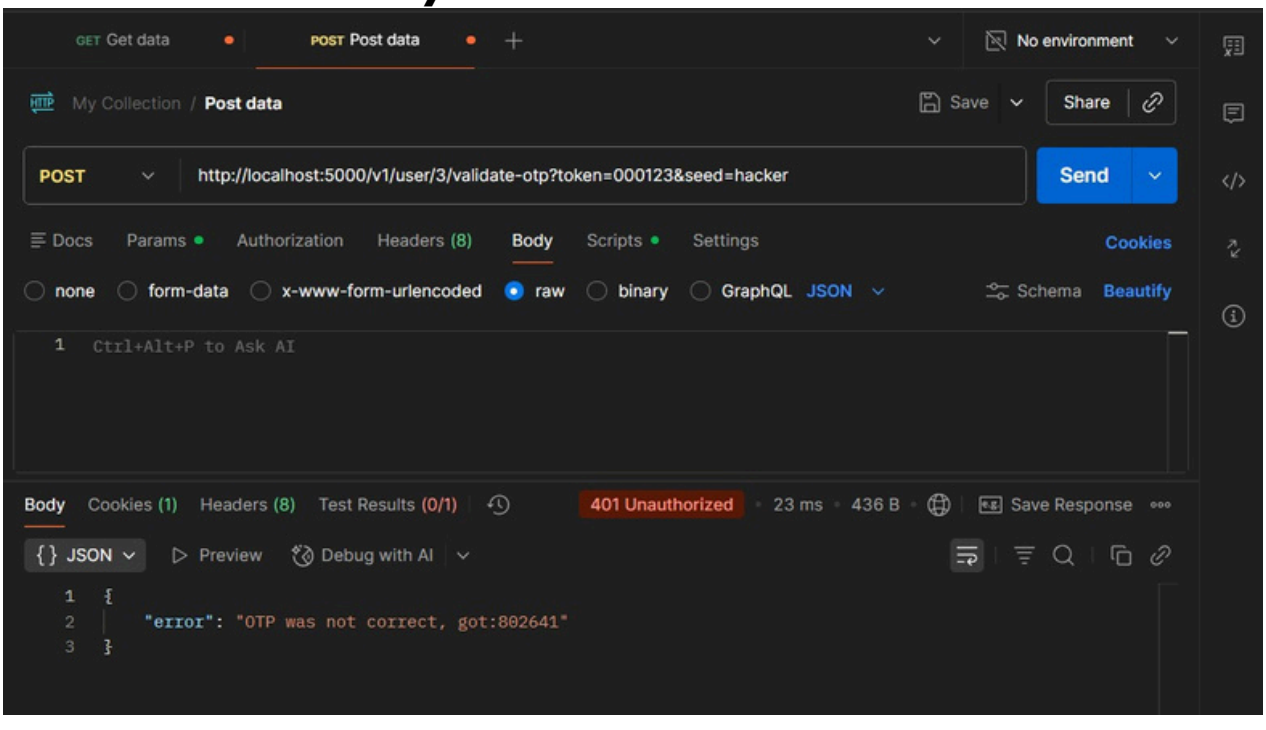
-Absence of input validation



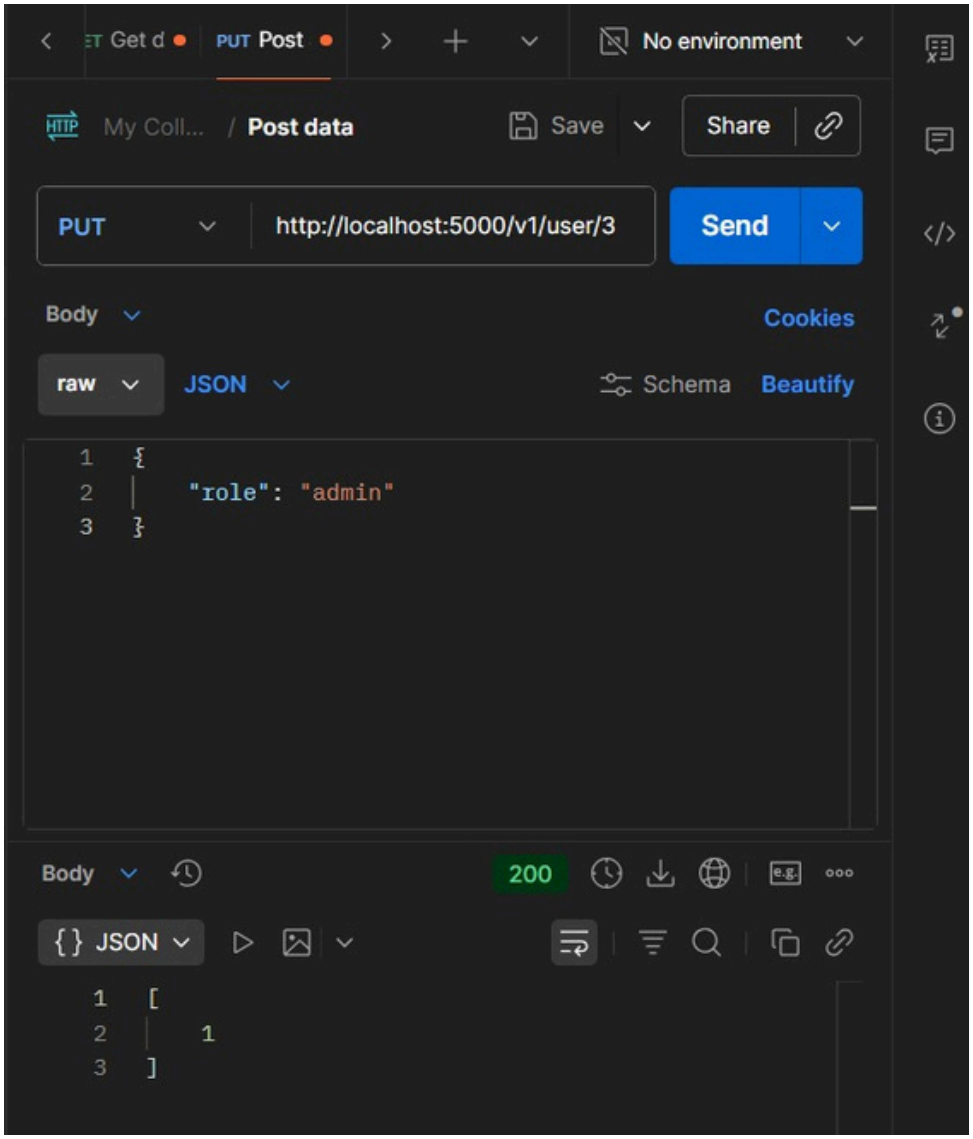
-Sql injection

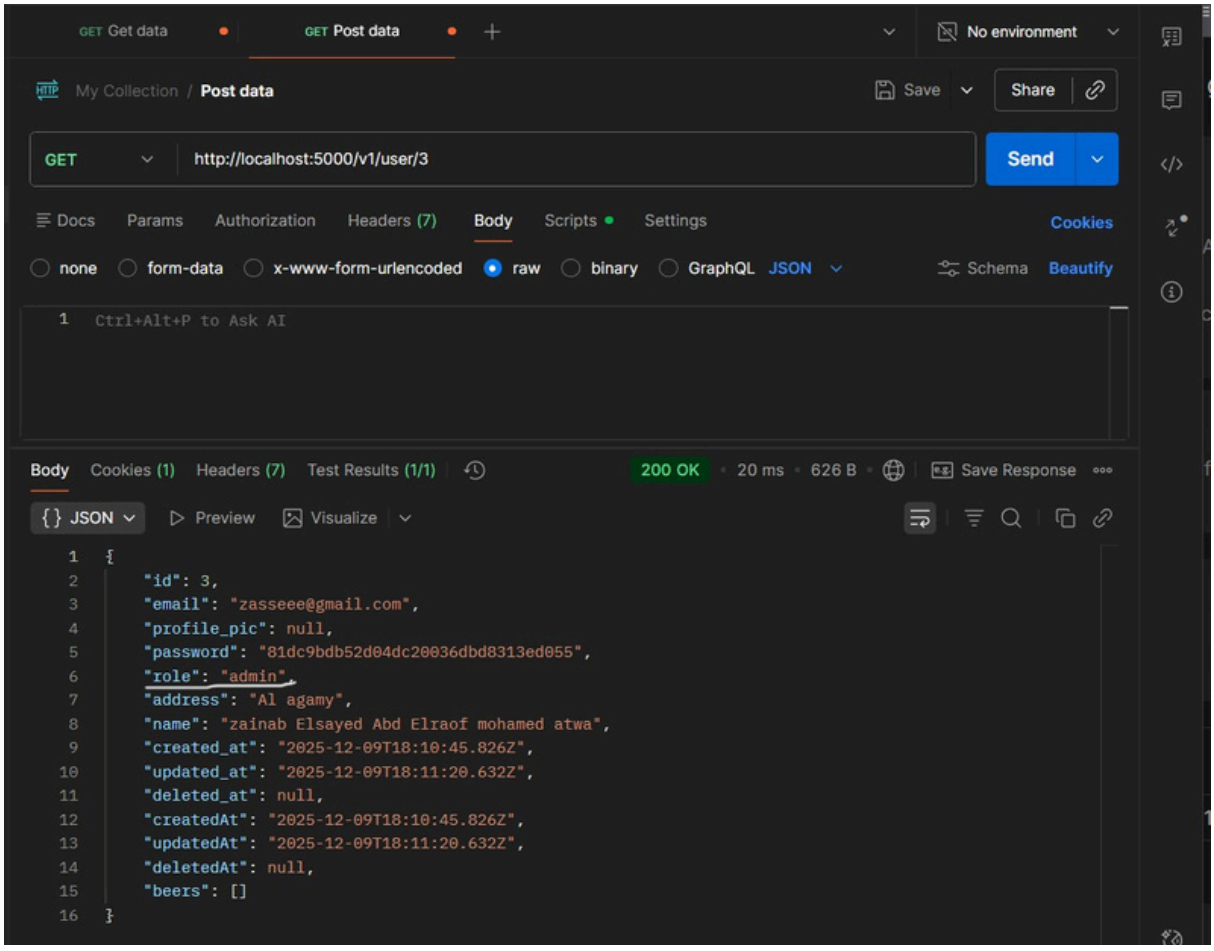


-OTP leak vulnerability



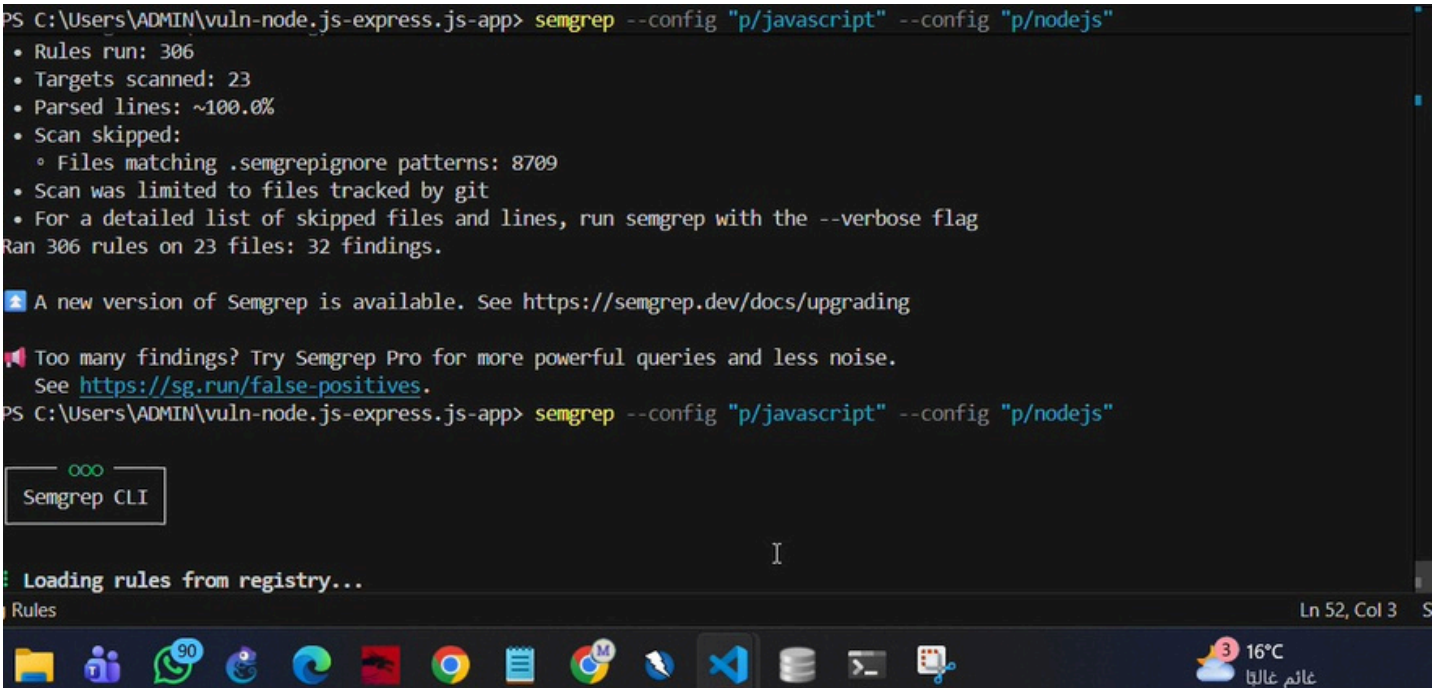
-privilage escalation(mass assignment)



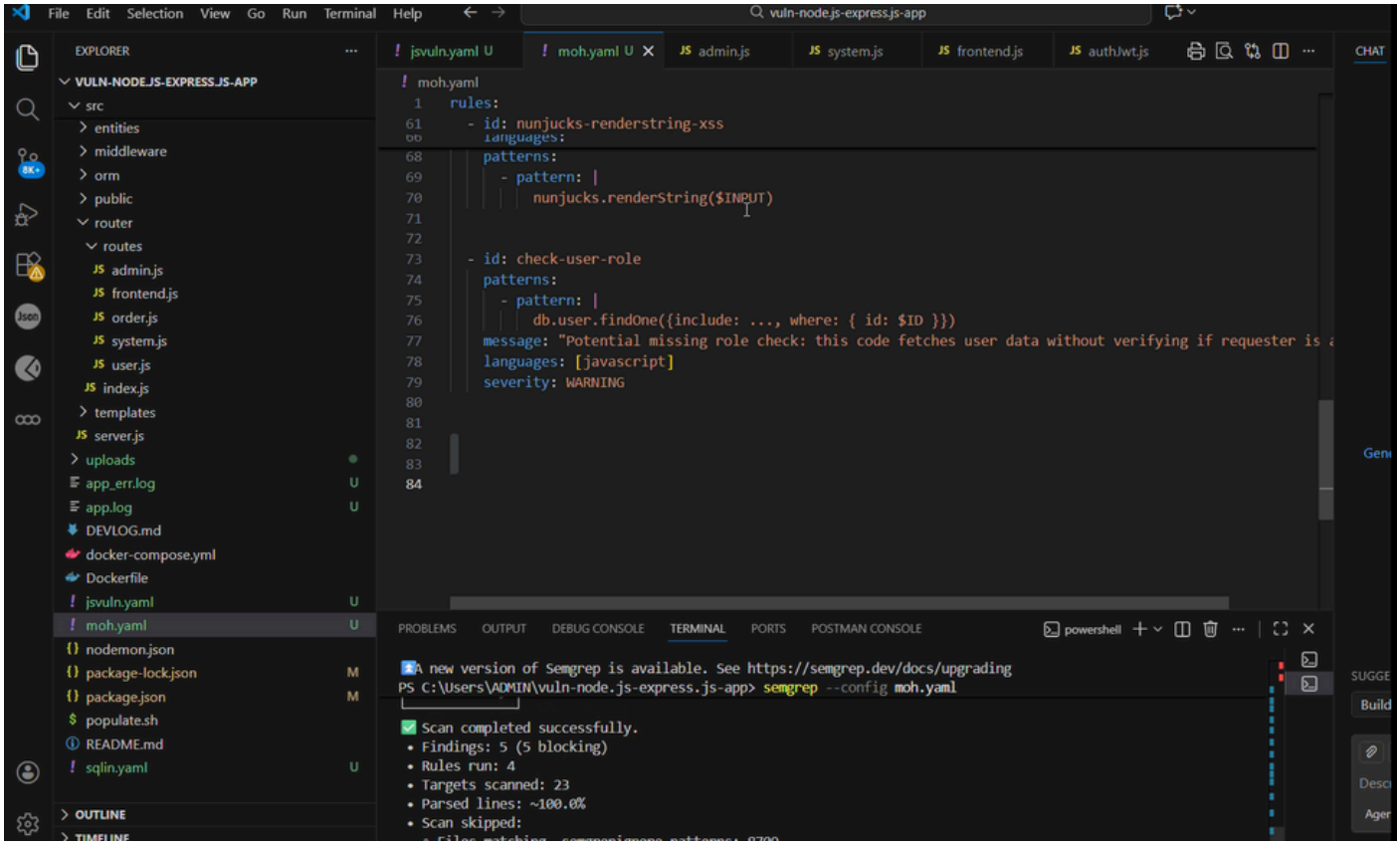


Section 2 - Semgrep (SAST) finding :

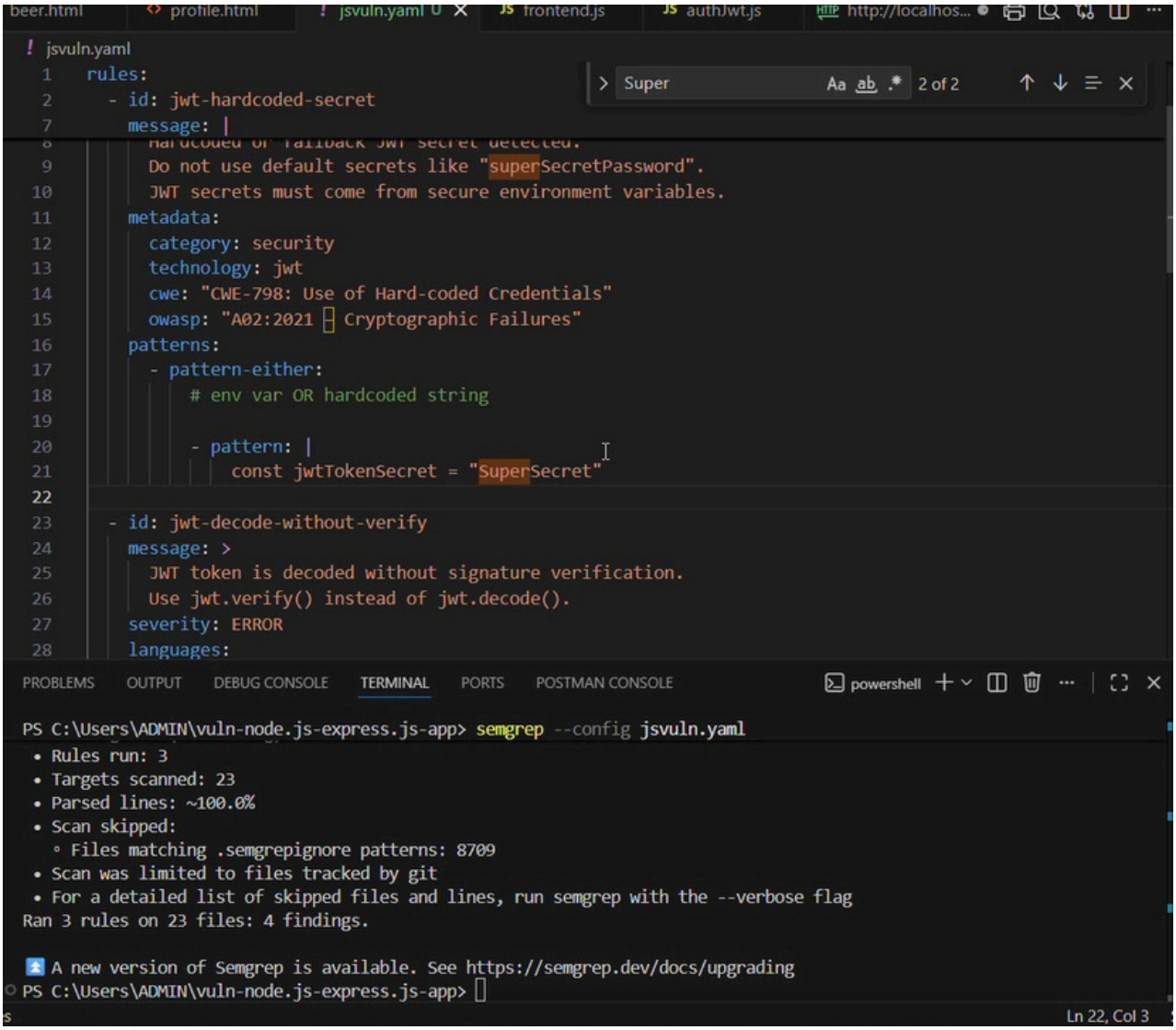
General Semgrep Rule :



- First semgrep Rule :



- Second semgrep Rule :

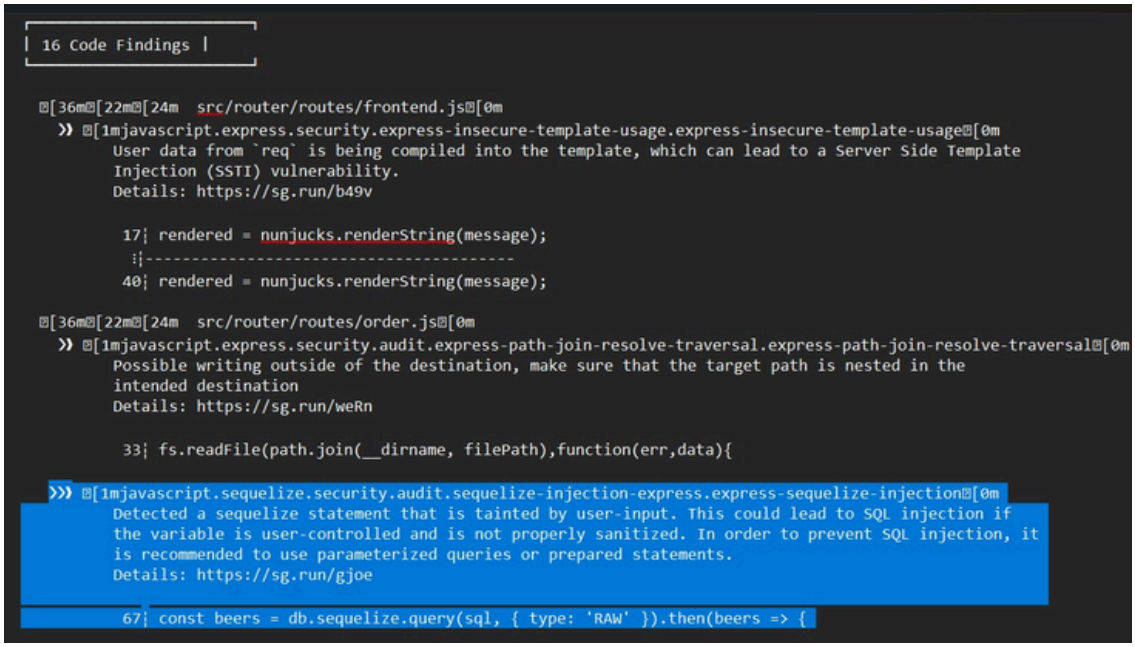


- Third semgrep Rule :

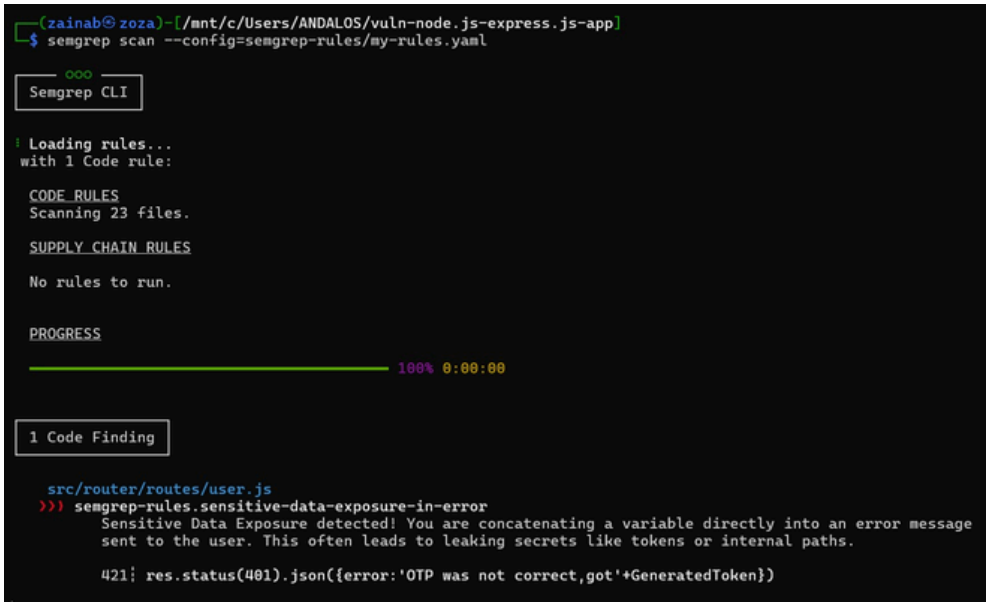
```yaml
rules:
  # 1. SQL Injection Rule (Regex)
  - id: force-sql-injection-detection
    languages:
      - javascript
    message: "SQL Injection detected! Found a SQL statement combined with '+' operator."
    severity: ERROR
    patterns:
      - pattern-regex: 'SELECT .* \+'

  # 2. Sensitive Data Exposure Rule (OTP Leak)
  - id: sensitive-data-exposure-in-error
    languages:
      - javascript
    message: "Sensitive Data Exposure detected! You are concatenating a variable directly into an error message sent to the user. This often leads to leaking secrets like
    severity: ERROR
    metadata:
      owasp: "A05: Security Misconfiguration"
    patterns:
      - pattern: |
          res.status(...).json({error: "..." + $VAR})

  # 3. Mass Assignment Rule (Privilege Escalation)
  - id: sequelize-mass-assignment
    languages:
      - javascript
    message: "Mass Assignment Vulnerability detected! Passing 'req.body' directly to the update function is dangerous. It allows attackers to inject restricted fields lik
    severity: ERROR
    metadata:
      owasp: "A01: Broken Access Control"
    patterns:
      - pattern: $MODEL.update(req.body, ...)
```

- Third semgrep Results :

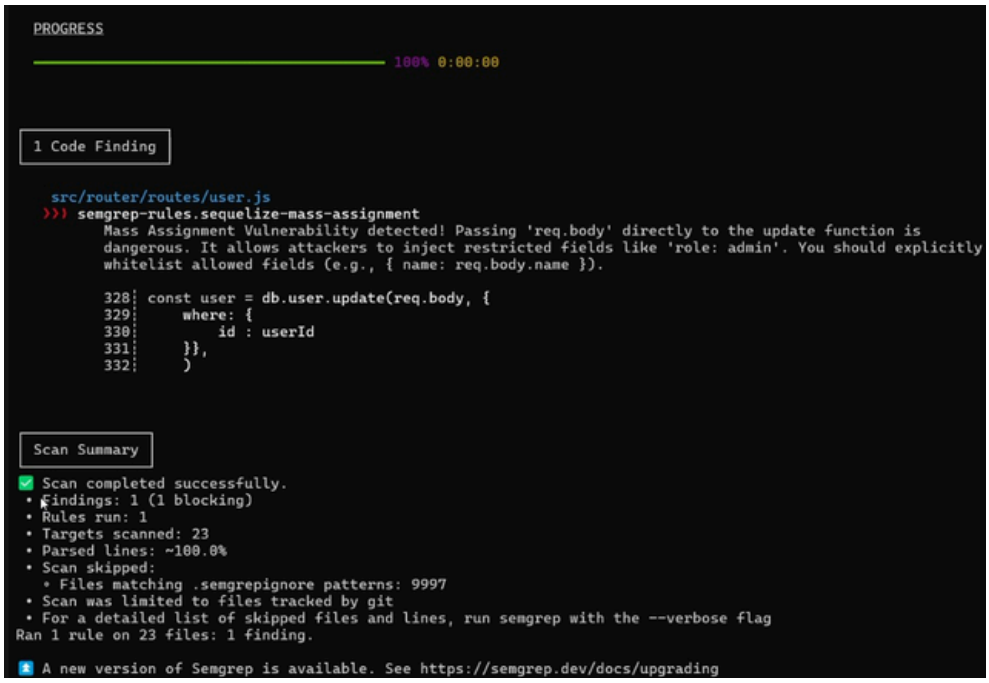
- SQL injection :



- OTP leak :



-privilage escalation :



Vulnerability Summary Table

Vuln ID	Endpoint / Feature	OWASP Category	File : Lines	Semgrep
V1	`GET /v1/user/{user_id}`	A01:2021 – Broken Access Control	`src/router/routes/user.js : 51-56`	Custom Rule
V2	`GET /?message=`	A03:2021 – Injection	`src/router/routes/frontend.js : 12-19`	Custom Rule
V3	`GET /v1/redirect/?url=`	A10:2017 – Unvalidated Redirects & Forwards	`src/router/routes/system.js : 33-44`	Custom Rule
V4	`POST /v1/admin/upload-pic`	A05:2021 – Security Misconfiguration	`src/router/routes/admin.js : 50-67`	Custom Rule
V5	`GET /v1/admin/users` – JWT Authentication (Token Verification Missing)	A02:2021 – Cryptographic Failures	`src/router/routes/user.js : 191-192`	Custom Rule
V6	`GET /v1/beer-pic` (Path Traversal)	A01:2021 – Broken Access Control	`/v1/beer-pic : 29-37`	Custom / General Rule
V7	`POST /v1/user/token` – JWT Implementation (Weak Secret Key)	A02:2021 – Cryptographic Failures	`src/router/routes/user.js : 18-415`	Custom / General Rule
V8	Email Input Validation (Missing / Improper)	A03:2021 – Injection	`src/template/user_register.html : 45`	Custom Rule
V9	`GET /v1/search?filter={query}`	A03:2021 – Injection	`src/router/routes/order.js : 65-70`	Custom Rule
V10	`POST /v1/user/{id}/validate-otp`	A05:2021 – Security Misconfiguration	`src/router/routes/user.js : 318-319`	Custom Rule
V11	`PUT /v1/user/{user_id}`	A01:2021 – Broken Access Control	`src/router/routes/user.js : 323-330`	Custom Rule

Section 3 - Fixes :

-Broken object level Authorization:

```
app.get('/v1/user/:id', (req, res) => {

// ✅ Role check (Admin فقط)
if (!req.session.user || req.session.user.role !== 'admin') {
  return res.status(403).json({ error: 'Forbidden: Admin only' });
}

db.user.findOne({
  where: { id: req.params.id }
})
.then(user => {
  res.json(user);
});

});
```

-Reflected xss:

```
app.get('/', (req, res) => {
console.log(req.session);

const message = req.query.message || "Please log in to continue";

// ✅ لا تستخدم renderString على input المستخدم
res.render('user.html', {
    message: message   // Nunjucks هيعمل escape تلقائي
});
});
```

-Open redirect:

```
app.get('/v1/redirect/', (req, res) => {
const allowedPaths = [
    'http://localhost:5000',
    'http://facebook.com',

];

const path = req.query.url;

if (!allowedPaths.includes(path)) {
    return res.status(400).json({ error: 'Invalid redirect path' });
}

res.redirect(path);
});
```

-Input invalidation (unstericted file upload) :

```
const path = require('path');
const multer = require('multer');

// تعريف multer مع fileFilter
const uploadImage = multer({
    dest: './uploads/',        // مكان حفظ الصور
    limits: { fileSize: 2 * 1024 * 1024 }, // 2 الحد الأقصى MB
    fileFilter: (req, file, cb) => {
        const allowedMimeTypes = ['image/jpeg', 'image/png', 'image/jpg', 'image/pjpeg'];
        const allowedExtensions = ['.jpg', '.jpeg', '.png'];

        const ext = path.extname(file.originalname).toLowerCase();

        if (allowedMimeTypes.includes(file.mimetype.toLowerCase()) && allowedExtensions.includes(ext)) {
            cb(null, true); // القبول بالملف
        } else {
            cb(new Error('Only image files are allowed'), false); // رفض أي ملف آخر
        }
    }
});

// route رفع الصور
app.post('/v1/admin/upload-pic/', (req, res) => {
    uploadImage.single('file')(req, res, function (err) {

        // خطأ من multer (نوع الملف / الحجم)
        if (err) {
            return res.status(400).json({ error: err.message });
        }

        // لو مفيش فايل
        if (!req.file) {
            return res.status(400).json({ error: 'No file uploaded or invalid file type' });
        }

        // نجاح
        res.status(200).json({ message: 'Image uploaded successfully' });
    });
});
```

-Weak secret:

```
     * @return {array<User>} 200 - success response - application/json
     */
    app.get('/v1/admin/users/', (req,res) =>{
        //console.log("auth",req.headers.authorization)
        if (req.headers.authorization){
        const user_object = jwt.verify(req.headers.authorization.split(' ')[1],config.jwtSecret)
        db.user.findAll({include: "beers"})
            .then((users) => {
                if (user_object.role =='admin'){
                    //console.log("fetch users")
                res.json(users);
                }
                else{
                res.json({error:"Not Admin, try again"})
            }

                return;
```

-Unverify jwt:

```
if(!req.query.id){ // if not provided take from session
    //if using jwt take from header:
    if(!req.session.user.id){
        //if not jwt found
        if(!req.headers.authorization){
            res.json({error:"Couldn't find user token"})
        }
        current_user_id = jwt.verify(req.headers.authorization.split(' ')[1]).id
    }
    current_user_id = req.session.user.id
}
current_user_id = req.query.id

const beer_id = req.params.beer_id;

db.beer.findOne({
    where:{id:beer_id}
}).then((beer) => {
```

-Traversal path:

```
app.get('/v1/beer-pic/', (req, res) => {
    const filename = req.query.picture;
    // 🔐 BLOCK PATH TRAVERSAL
    if (!filename) {
        return res.status(403).json({ error: 'Bad ' });
    }

    const uploadsDir = path.resolve(__dirname, '../../../uploads');
    const requestedPath = path.resolve(uploadsDir, filename);

    // 🔐 BLOCK PATH TRAVERSAL
    if (!requestedPath.startsWith(uploadsDir)) {
        return res.status(403).json({ error: 'Access denied' });
    }

    fs.readFile(requestedPath, (err, data) => {
        if (err) {
```

-Absence of input validation

```
<h4 class="mb-2">Welcome to Sneat! 👋</h4>
<p class="mb-4">{{message | safe}}</p>

<form id="formAuthentication" class="mb-3" action="/registerform" method
  <div class="mb-3">
    <label for="email" class="form-label">Email or Username</label>
    <input
      type="email"
      class="form-control"
      id="email"
      name="email"
      placeholder="Enter your email or username"
      autofocus
    />
  </div>
  <div class="mb-3 form-password-toggle">
    <div class="d-flex justify-content-between">
      <label class="form-label" for="password">Password</label>
```

MS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE powershell

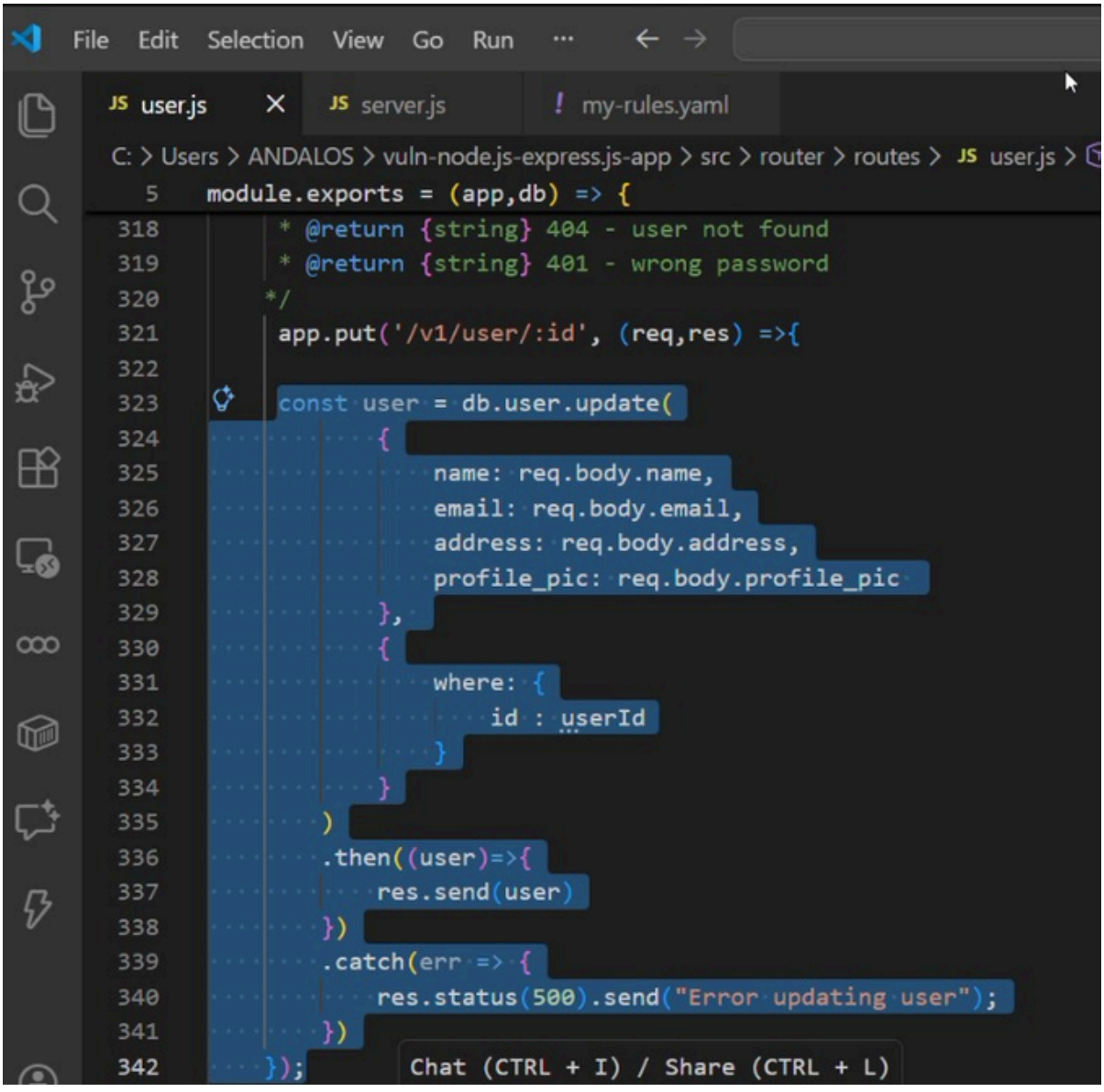
-Sql injection

```
66          // 1. Whitelist Validation
67          const allowedFilters = ['name', 'price', 'id', 'currency', 'stock'];
68          if (!allowedFilters.includes(filter)) {
69              return res.status(400).send("Invalid filter column");
70          }
71
72          // 2. Parameterization
73          const sql = `SELECT * FROM beers WHERE ${filter} = :queryVal`;
74
75          const beers = db.sequelize.query(sql, {
76              replacements: { queryVal: query }, // Sequelize will handle escaping
77              type: db.sequelize.QueryTypes.SELECT
78          }).then(beers => {
79              res.status(200).send(beers);
80
81          }).catch(function (err) {
82              // Handle error
83              console.error(err);
84              res.status(500).send("Error executing query");
85          })
```

-OTP leak

```
415                 }
416                 if(req.query.seed){
417                  req.session.otp = GeneratedToken // add generated token to session
418                  req.session.save(function(err) {
419                      // session saved
420                   })
421                  res.status(401).json({error:'OTP was not correct'})
422                  return;
423                 }
424                 res.status(401).json({error:'OTP was not correct'})
425             })
426     });
427 };
```

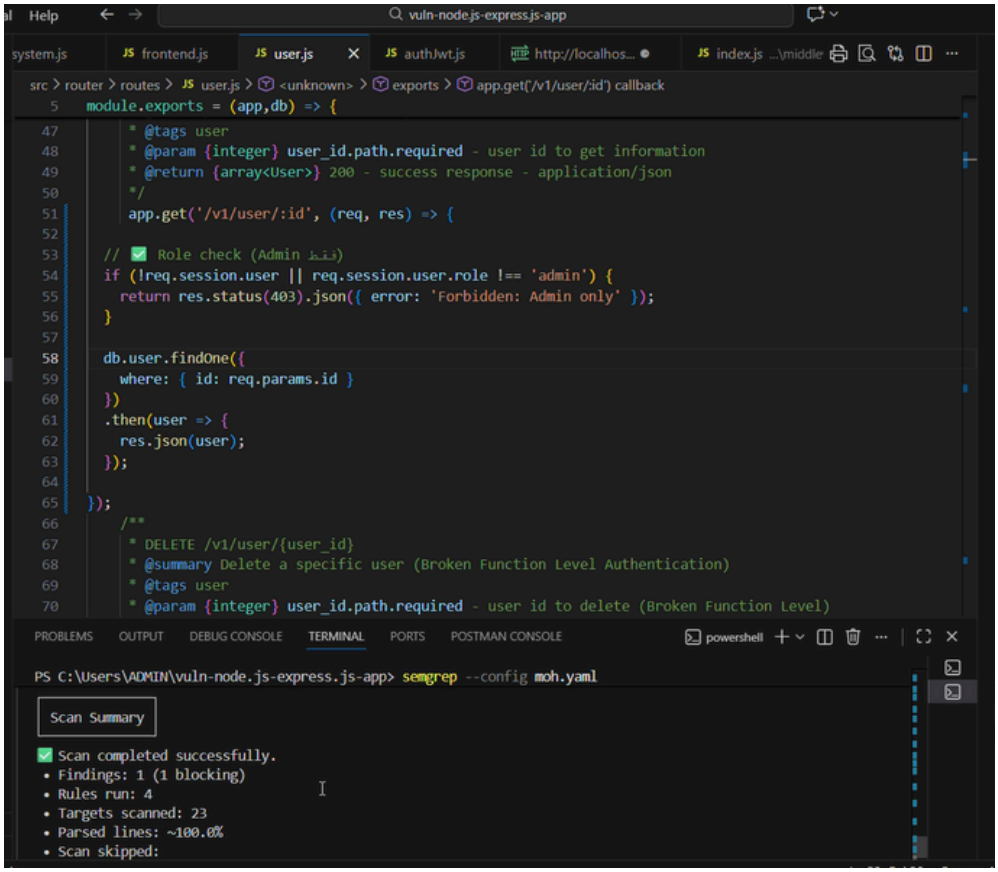
-privilage escalation



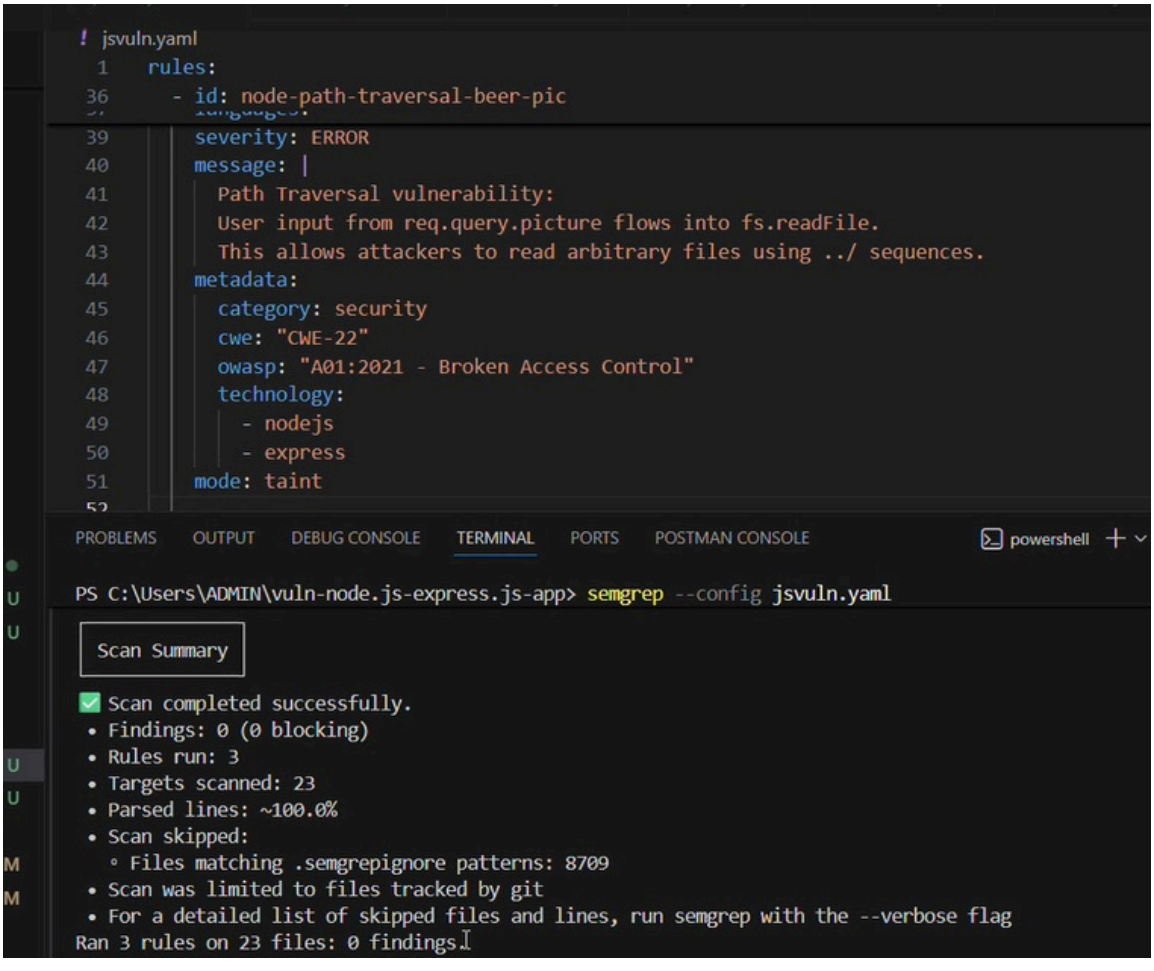
Section 4 - Re-test E vidence :

-Semgrep result after fix code :

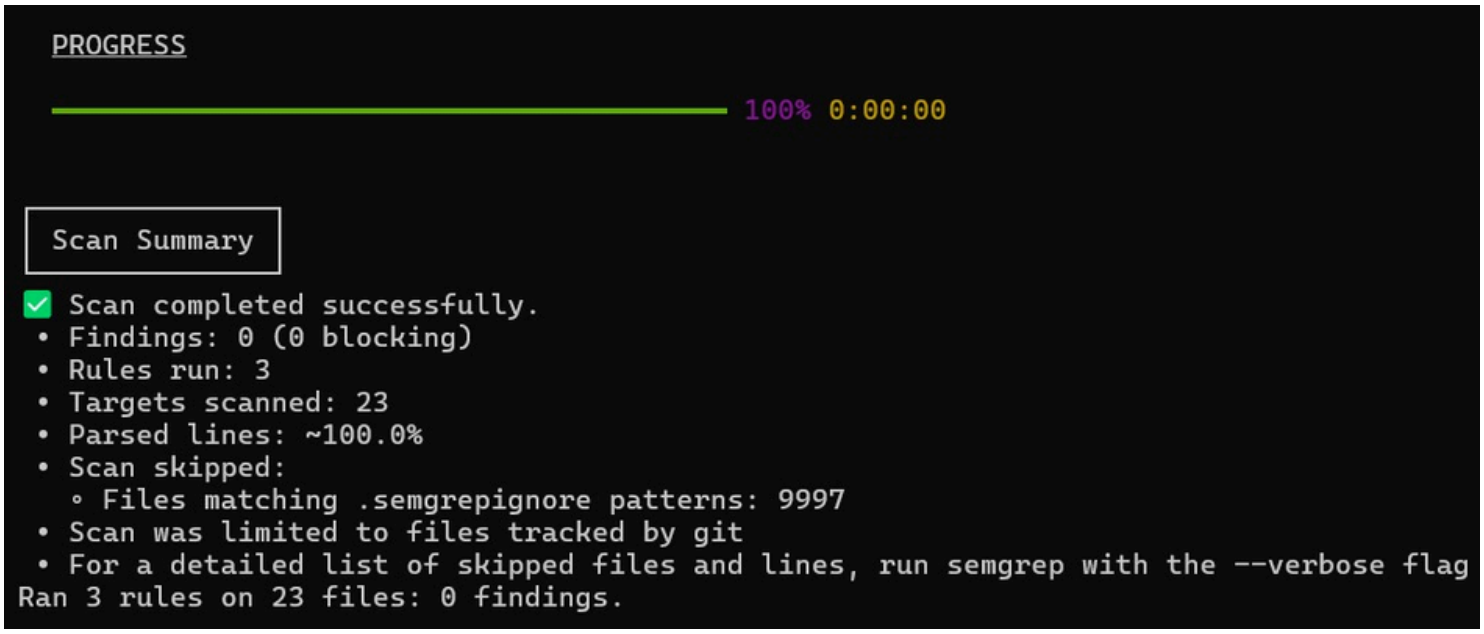
- First Semgrep rule



- Second Semgrep rule



- Third Semgrep rule



- Summary:

-The vulnerabilities have been closed :

1-Broken access control

2-Injection

3-Security misconfigration

4-Cryptographic failures

5-Invalidate redirect forward

-Tools have been used:

1-owasp zap

2-postman

3-semgrep

4-www.jwt.io