Verbose error messages

A05:2021-Security Misconfiguration

Security Misconfiguration meaning

Security features exist but are configured badly

Routes and files

Files

- frontend.js
- order.js

Routes

- /
- /login
- /registerform
- /v1/search/:filter/:query

vulnerable code

```
const beers = db.sequelize.query(sql, { type: 'RAW' }).then(beers => {
    res.status(200).send(beers);
}).catch(function (err) {
    res.status(501).send("error, query failed: "+err)
  })
```

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

```
(e) =>
{
    console.log(e)
    res.redirect('/?message=Error registering,
```

```
console.log(userEmail)
console.log(emailExpression.test(userEmail))
```

```
var userPassword = req.query.password;
console.log(req.query.message)
const user = db.user.findAll({
```

exploitation

http://localhost:5000/v1/search/invalid_column/1

using postman you could trigger an error:
(error, query failed: SequelizeDatabaseError: SQLITE_ERROR: no such column: invalid_column)

---

explanation

The application exposes excessive debug and error information in several places. Internal session data, exception details, database errors, and system-level failures are logged or displayed without proper restriction, which could reveal sensitive information about the application’s internal behavior.

Semgrep rule

```yaml
rules:
  - id: a05-verbose-errors
    message: A05 - Verbose error messages
    severity: MEDIUM
    languages: [javascript]
    pattern-either:
      - pattern: 'res.send( ... + $VAR + ... )'
      - pattern: 'console.log( ... )'

```

fixes

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

```
(e) =>
{
    console.log(e)
    res.redirect('/?message=Error registering,
```

```
console.log(userEmail)
console.log(emailExpression.test(userEmail))
```

```
var userPassword = req.query.password;
console.log(req.query.message)
const user = db.user.findAll({
```

Debug logging such as console.log() should be strictly limited to development and testing phases. All debug statements that expose session data, user input, credentials, or internal application state must be removed before deploying the application to production. Leaving such logs in production can leak sensitive information and assist attackers during reconnaissance.

fixes

```
res.status(501).send("error, query failed: "+err)
```

```
res.status(500).json({error: "An error occurred processing your request"})
```