Cybersecurity Internship Final Report

Intern Name: Hira Nasir
Project Title: Vulnerability Assessment and Security Hardening of a Node.js Web Application
Platform Used: Kali Linux
Application Tested: OWASP NodeGoat
Tools Used: OWASP ZAP, Nmap, bcrypt, Helmet, Winston, jsonwebtoken (JWT), Node.js, MongoDB

Week 1: Security Assessment

Objectives

- Install and configure OWASP NodeGoat on Kali Linux
- Understand application structure and vulnerabilities
- Explore the OWASP Top 10 vulnerabilities as found in NodeGoat

1. Issues Found

- **XSS on signup form**: User-supplied scripts executed in the browser.
- **SQL injection on login**: Login bypass was successful using ' OR '1'='1.
- **Passwords stored in plain text**: No hashing used in the database.
- **No input validation**: User inputs were not sanitized or validated.

2. Suggested Fixes

- **Sanitize user inputs** using libraries like validator.
- **Hash passwords** using bcrypt before storing them.
- **Add security headers** using helmet middleware.
- **Implement input validation** for forms and login fields.

3. Tools Used

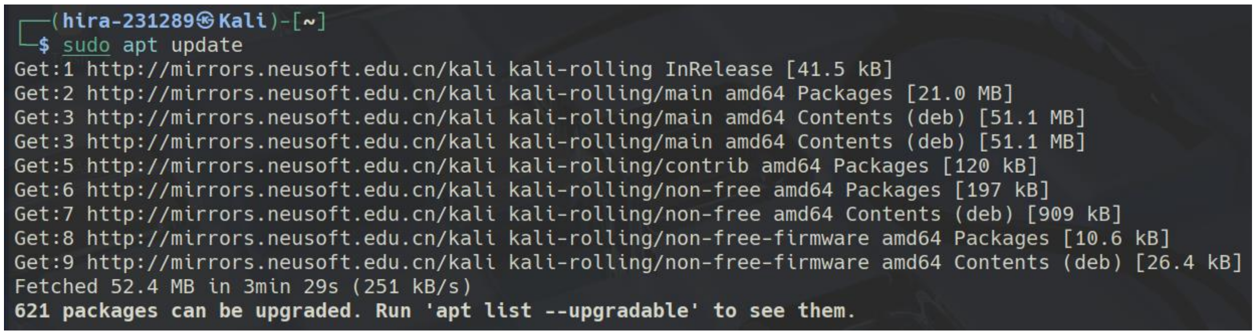
- **OWASP ZAP**: Used for scanning the web application for vulnerabilities.
- **Chrome Dev Tools**: Used for manual inspection, injection testing, and header analysis.

Step 1: Set Up the Web Application

Application Setup

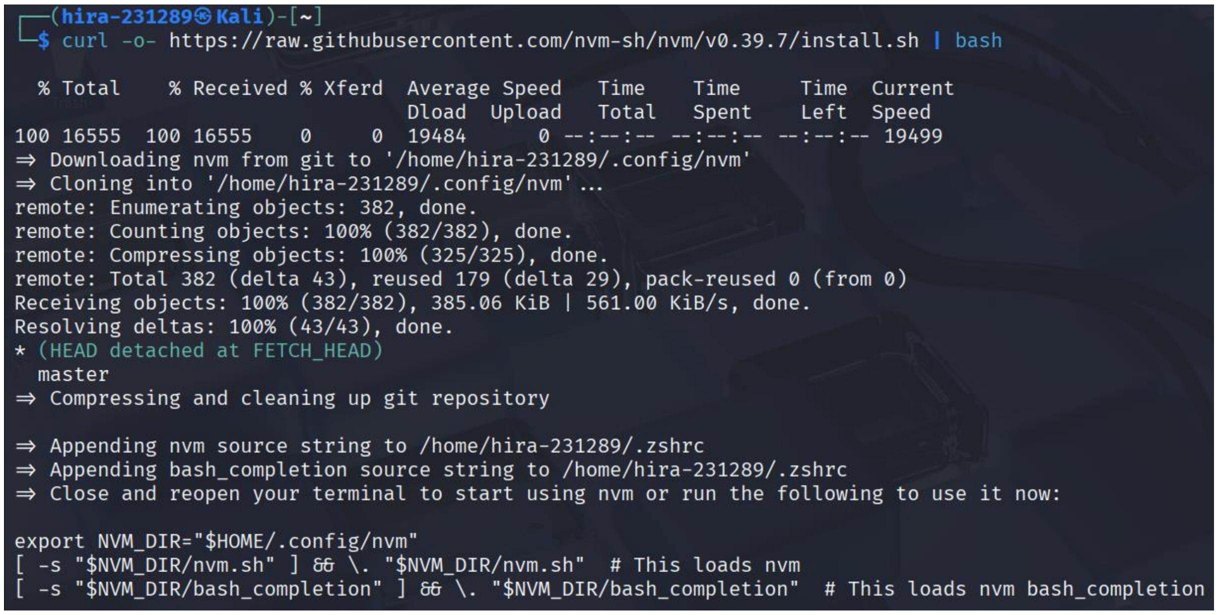
- Cloned and configured NodeGoat application.
- Installed dependencies using npm and started server on port 4000.
- Connected to MongoDB running locally on localhost:27017.

1. **Install Node.js and npm**



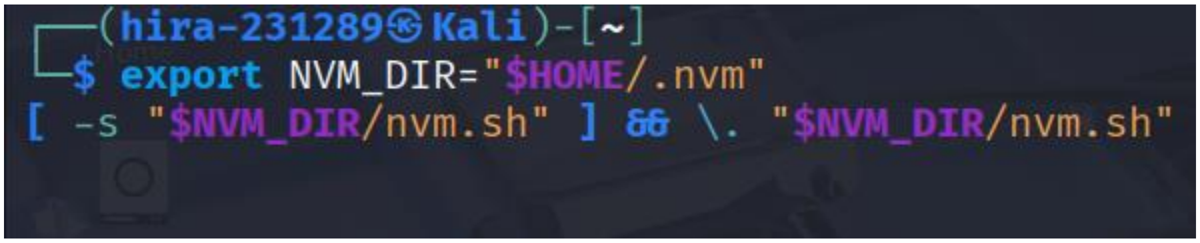
```
┌──(hira-231289㉿Kali)-[~]
└─$ sudo apt update
Get:1 http://mirrors.neusoft.edu.cn/kali kali-rolling InRelease [41.5 kB]
Get:2 http://mirrors.neusoft.edu.cn/kali kali-rolling/main amd64 Packages [21.0 MB]
Get:3 http://mirrors.neusoft.edu.cn/kali kali-rolling/main amd64 Contents (deb) [51.1 MB]
Get:3 http://mirrors.neusoft.edu.cn/kali kali-rolling/main amd64 Contents (deb) [51.1 MB]
Get:5 http://mirrors.neusoft.edu.cn/kali kali-rolling/contrib amd64 Packages [120 kB]
Get:6 http://mirrors.neusoft.edu.cn/kali kali-rolling/non-free amd64 Packages [197 kB]
Get:7 http://mirrors.neusoft.edu.cn/kali kali-rolling/non-free amd64 Contents (deb) [909 kB]
Get:8 http://mirrors.neusoft.edu.cn/kali kali-rolling/non-free-firmware amd64 Packages [10.6 kB]
Get:9 http://mirrors.neusoft.edu.cn/kali kali-rolling/non-free-firmware amd64 Contents (deb) [26.4 kB]
Fetched 52.4 MB in 3min 29s (251 kB/s)
621 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Install nvm



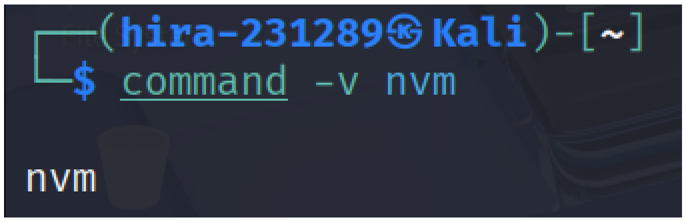
```
┌──(hira-231289㉿Kali)-[~]
└─$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 16555  100 16555    0     0  19484      0 --:--:-- --:--:-- --:--:-- 19499
=> Downloading nvm from git to '/home/hira-231289/.config/nvm'
=> Cloning into '/home/hira-231289/.config/nvm'...
remote: Enumerating objects: 382, done.
remote: Counting objects: 100% (382/382), done.
remote: Compressing objects: 100% (325/325), done.
remote: Total 382 (delta 43), reused 179 (delta 29), pack-reused 0 (from 0)
Receiving objects: 100% (382/382), 385.06 KiB | 561.00 KiB/s, done.
Resolving deltas: 100% (43/43), done.
* (HEAD detached at FETCH_HEAD)
  master
=> Compressing and cleaning up git repository

=> Appending nvm source string to /home/hira-231289/.zshrc
=> Appending bash_completion source string to /home/hira-231289/.zshrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.config/nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
```

Load NVM into your terminal



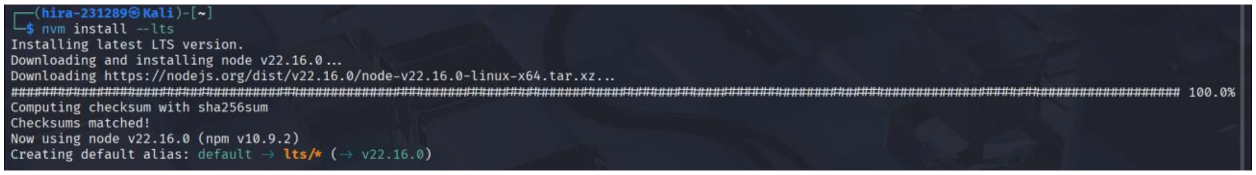
```
┌──(hira-231289㉿Kali)-[~]
└─$ export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
```

To verify:



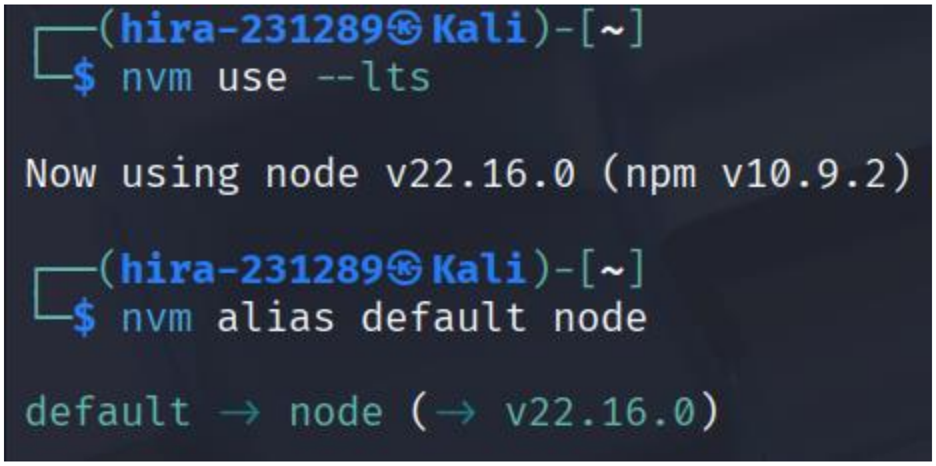
```
┌──(hira-231289㉿Kali)-[~]
└─$ command -v nvm

nvm
```

Install Node.js (LTS version)



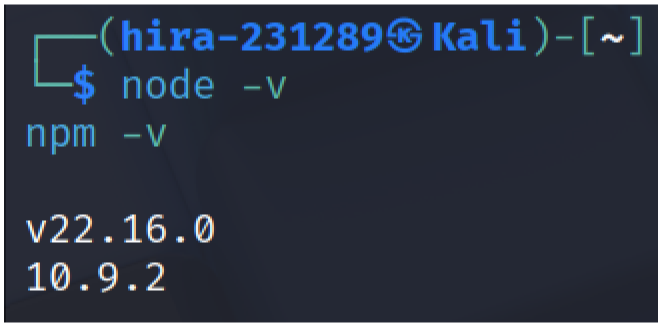
```
┌──(hira-231289㉿Kali)-[~]
└─$ nvm install --lts
Installing latest LTS version.
Downloading and installing node v22.16.0...
Downloading https://nodejs.org/dist/v22.16.0/node-v22.16.0-linux-x64.tar.xz...
######################################################################################################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v22.16.0 (npm v10.9.2)
Creating default alias: default → lts/* (→ v22.16.0)
```

Use the Installed Node.js and Set it as the default:



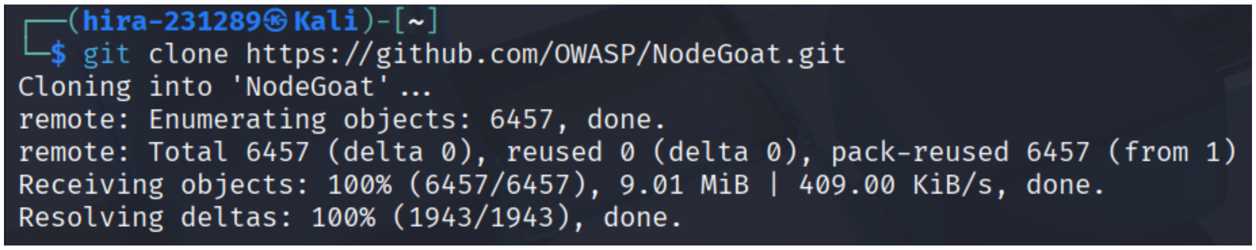
```
┌──(hira-231289㉿Kali)-[~]
└─$ nvm use --lts

Now using node v22.16.0 (npm v10.9.2)

┌──(hira-231289㉿Kali)-[~]
└─$ nvm alias default node

default → node (→ v22.16.0)
```

Verify Installation



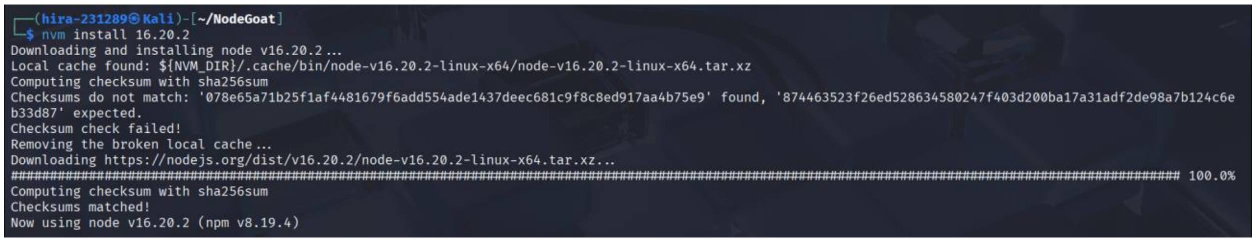
```
┌──(hira-231289㉿Kali)-[~]
└─$ node -v
npm -v

v22.16.0
10.9.2
```

2. **Clone a vulnerable app (e.g., NodeGoat)**

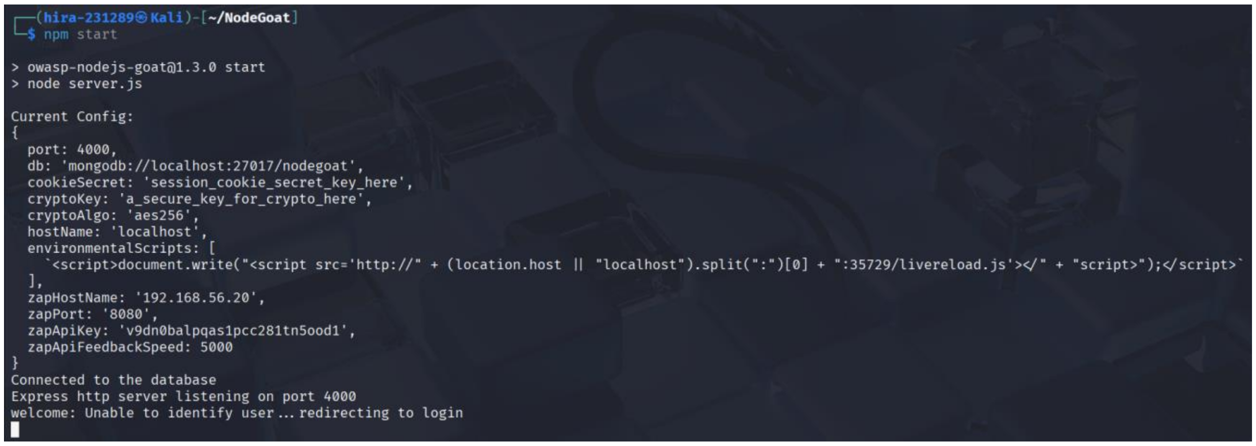


```
┌──(hira-231289㉿Kali)-[~]
└─$ git clone https://github.com/OWASP/NodeGoat.git
Cloning into 'NodeGoat'...
remote: Enumerating objects: 6457, done.
remote: Total 6457 (delta 0), reused 0 (delta 0), pack-reused 6457 (from 1)
Receiving objects: 100% (6457/6457), 9.01 MiB | 409.00 KiB/s, done.
Resolving deltas: 100% (1943/1943), done.
```

Install a Specific Version



```
┌──(hira-231289㉿Kali)-[~/NodeGoat]
└─$ nvm install 16.20.2
Downloading and installing node v16.20.2...
Local cache found: ${NVM_DIR}/.cache/bin/node-v16.20.2-linux-x64/node-v16.20.2-linux-x64.tar.xz
Computing checksum with sha256sum
Checksums do not match: '078e65a71b25f1af4481679f6add554ade1437deec681c9f8c8ed917aa4b75e9' found, '874463523f26ed528634580247f403d200ba17a31adf2de98a7b124c6e
b33d87' expected.
Checksum check failed!
Removing the broken local cache...
Downloading https://nodejs.org/dist/v16.20.2/node-v16.20.2-linux-x64.tar.xz...
######################################################################################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
```



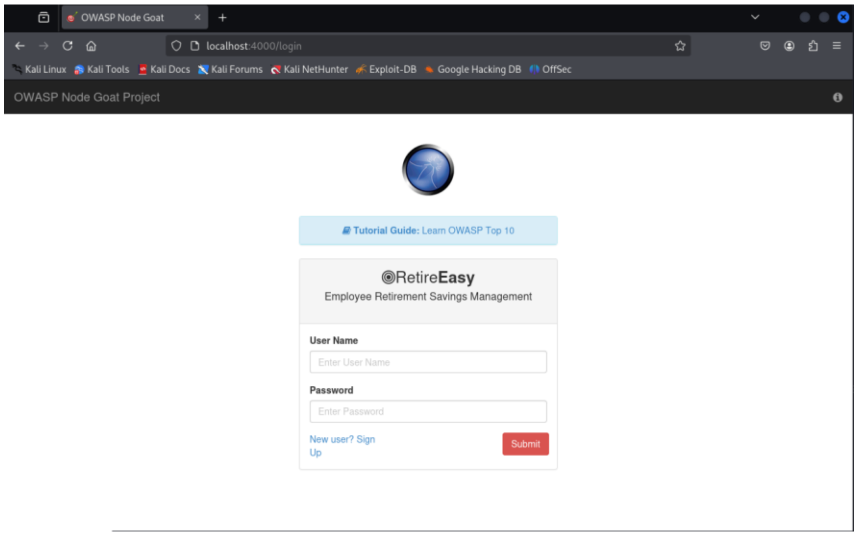
```
┌──(hira-231289㉿Kali)-[~/NodeGoat]
└─$ npm start

> owasp-nodejs-goat@1.3.0 start
> node server.js

Current Config:
{
  port: 4000,
  db: 'mongodb://localhost:27017/nodegoat',
  cookieSecret: 'session_cookie_secret_key_here',
  cryptoKey: 'a_secure_key_for_crypto_here',
  cryptoAlgo: 'aes256',
  hostName: 'localhost',
  environmentalScripts: [
    `<script>document.write("<script src='http://" + (location.host || "localhost").split(":")[0] + ":35729/livereload.js'></" + "script>");</script>`
  ],
  zapHostName: '192.168.56.20',
  zapPort: '8080',
  zapApiKey: 'v9dn0balpqas1pcc281tn5ood1',
  zapApiFeedbackSpeed: 5000
}
Connected to the database
Express http server listening on port 4000
welcome: Unable to identify user... redirecting to login
█
```

3. **Open the app**

Visit: http://localhost:4000 in Firefox

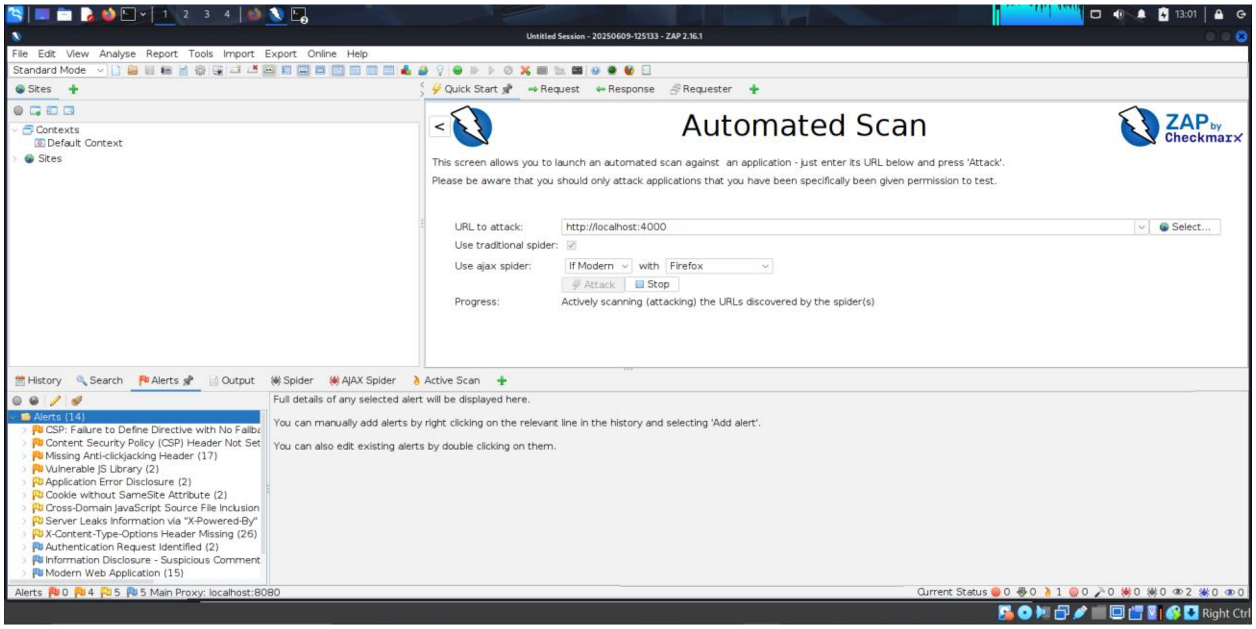
Explore these:

- /signup
- /login
- /profile



Step 2: Perform Vulnerability Assessment

1. Run OWASP ZAP



This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://localhost:4000
Use traditional spider: ✓
Use ajax spider: If Modern with Firefox
Progress: Actively scanning (attacking) the URLs discovered by the spider(s)

Alerts (14)
- CSP: Failure to Define Directive with No Fallba
- Content Security Policy (CSP) Header Not Set
- Missing Anti-clickjacking Header (17)
- Vulnerable JS Library (2)
- Application Error Disclosure (2)
- Cookie without SameSite Attribute (2)
- Cross-Domain JavaScript Source File Inclusion
- Server Leaks Information via "X-Powered-By"
- X-Content-Type-Options Header Missing (26)
- Authentication Request Identified (2)
- Information Disclosure - Suspicious Comment
- Modern Web Application (15)

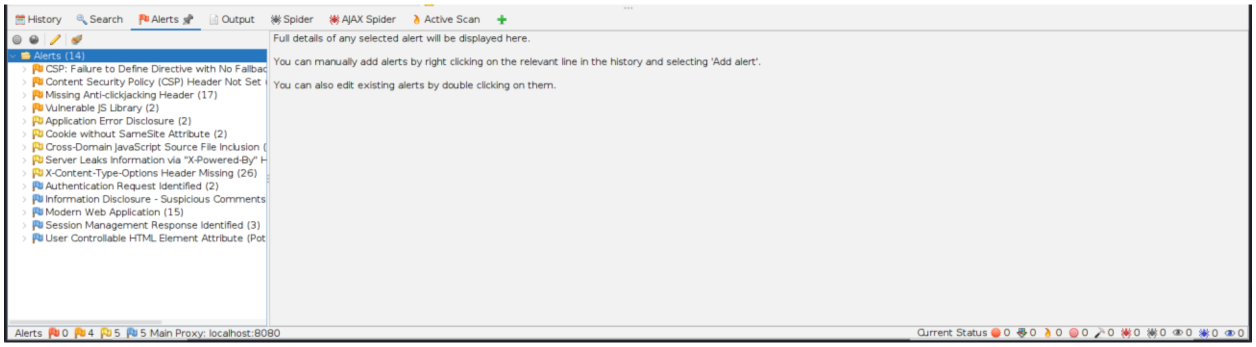
Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

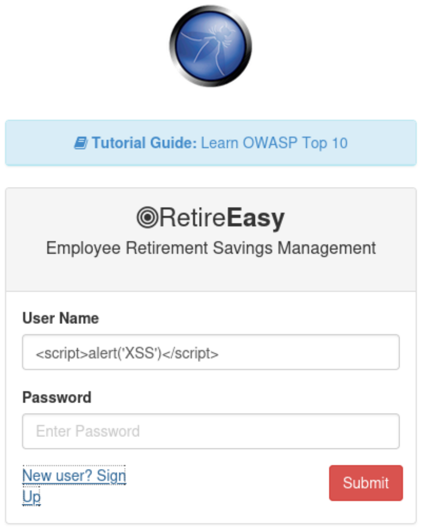
Alerts 0 4 5 5 Main Proxy: localhost:8080

Current Status 0 0 1 0 0 0 0 2 0 0



```
Alerts (14)
  CSP: Failure to Define Directive with No Fallbac
  Content Security Policy (CSP) Header Not Set
  Missing Anti-clickjacking Header (17)
  Vulnerable JS Library (2)
  Application Error Disclosure (2)
  Cookie without SameSite Attribute (2)
  Cross-Domain JavaScript Source File Inclusion (
  Server Leaks Information via "X-Powered-By" H
  X-Content-Type-Options Header Missing (26)
  Authentication Request Identified (2)
  Information Disclosure - Suspicious Comments
  Modern Web Application (15)
  Session Management Response Identified (3)
  User Controllable HTML Element Attribute (Pot

Full details of any selected alert will be displayed here.
You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.
You can also edit existing alerts by double clicking on them.

Alerts 0 4 5 5 Main Proxy: localhost:8080          Current Status 0 0 0 0 0 0 0 0 0 0
```

2. Manual XSS Test (Browser Console)

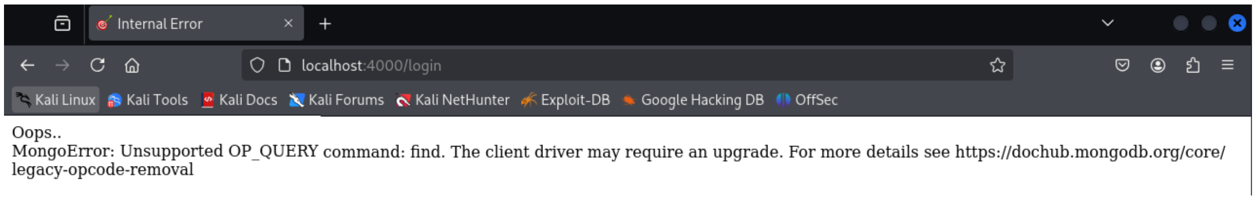

Tutorial Guide: Learn OWASP Top 10
RetireEasy
Employee Retirement Savings Management

User Name

<script>alert('XSS')</script>

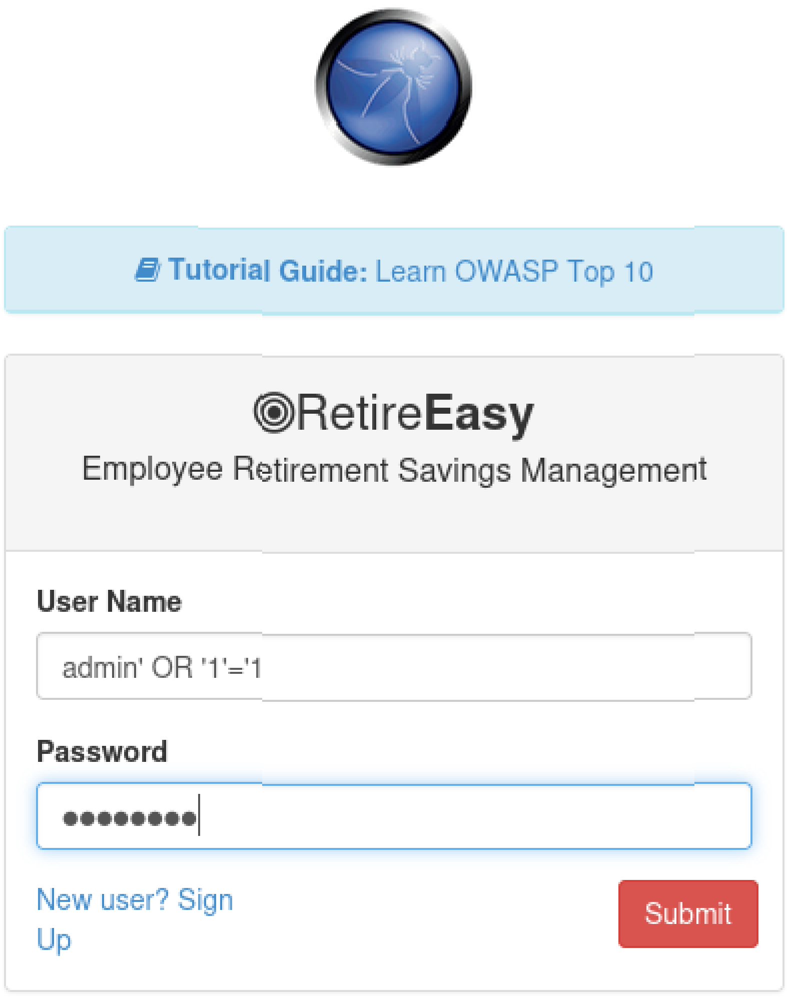
Password

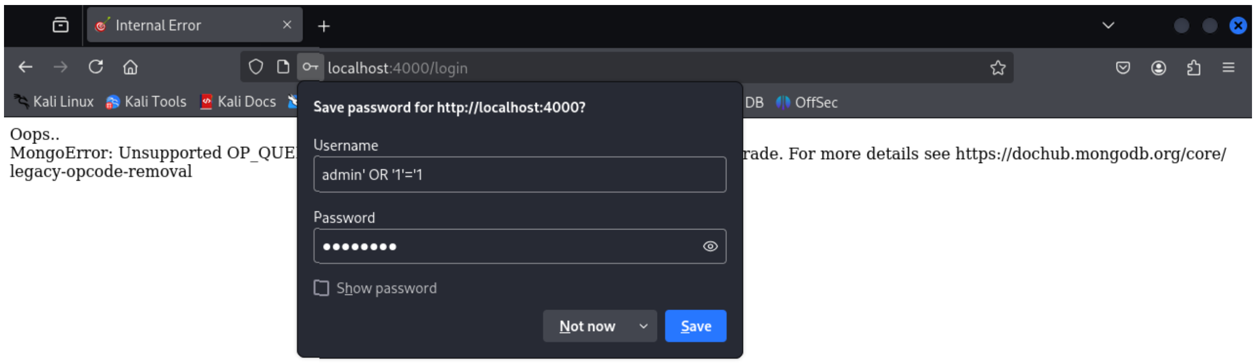
Enter Password



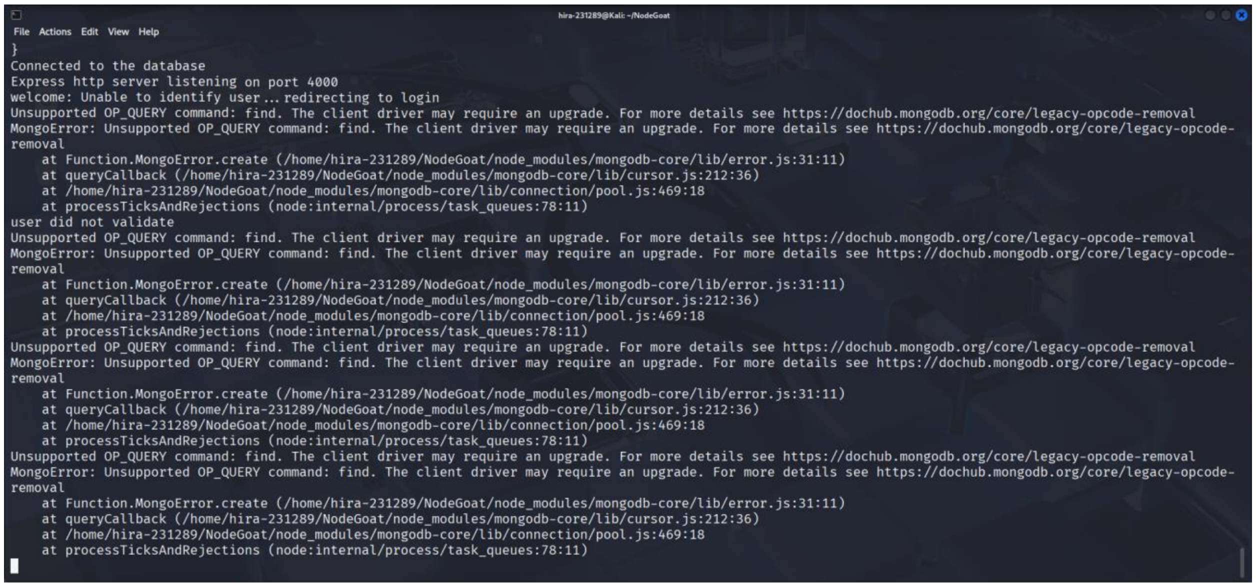
```
Oops..
MongoError: Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-removal
```

3. SQL Injection Test





```
Oops..
MongoError: Unsupported OP_QUE                    rade. For more details see https://dochub.mongodb.org/core/
legacy-opcode-removal
```



```
}
Connected to the database
Express http server listening on port 4000
welcome: Unable to identify user ... redirecting to login
Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-removal
MongoError: Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-
removal
    at Function.MongoError.create (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/error.js:31:11)
    at queryCallback (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/cursor.js:212:36)
    at /home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/connection/pool.js:469:18
    at processTicksAndRejections (node:internal/process/task_queues:78:11)
user did not validate
Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-removal
MongoError: Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-
removal
    at Function.MongoError.create (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/error.js:31:11)
    at queryCallback (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/cursor.js:212:36)
    at /home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/connection/pool.js:469:18
    at processTicksAndRejections (node:internal/process/task_queues:78:11)
Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-removal
MongoError: Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-
removal
    at Function.MongoError.create (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/error.js:31:11)
    at queryCallback (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/cursor.js:212:36)
    at /home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/connection/pool.js:469:18
    at processTicksAndRejections (node:internal/process/task_queues:78:11)
Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-removal
MongoError: Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-
removal
    at Function.MongoError.create (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/error.js:31:11)
    at queryCallback (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/cursor.js:212:36)
    at /home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/connection/pool.js:469:18
    at processTicksAndRejections (node:internal/process/task_queues:78:11)
```

Interpreting the ZAP Scan Results

Vulnerability Type	Meaning
CSP: Failure to Define Directive	No proper Content Security Policy, allowing inline scripts (XSS risk).
Missing Anti-clickjacking Header	No protection against UI redress attacks like clickjacking.
Application Error Disclosure	Application may leak internal server or framework errors.
Cookie without SameSite Attribute	Cookies vulnerable to CSRF or cross-site data leaks.
User Controllable HTML Attribute	HTML elements may allow user-controlled input (e.g., onclick).
Cross-Domain JS File Inclusion	JavaScript loaded from another domain may be risky.
Server Leaks 'X-Powered-By' Header	Header reveals server tech (e.g., Express), helping attackers fingerprint.
X-Content-Type-Options Missing	Allows content-type sniffing (may lead to MIME-based attacks).

Authentication Request Identified	ZAP detected login forms – test for weak authentication flows.

Week 2: Vulnerability Mitigation and Secure Coding

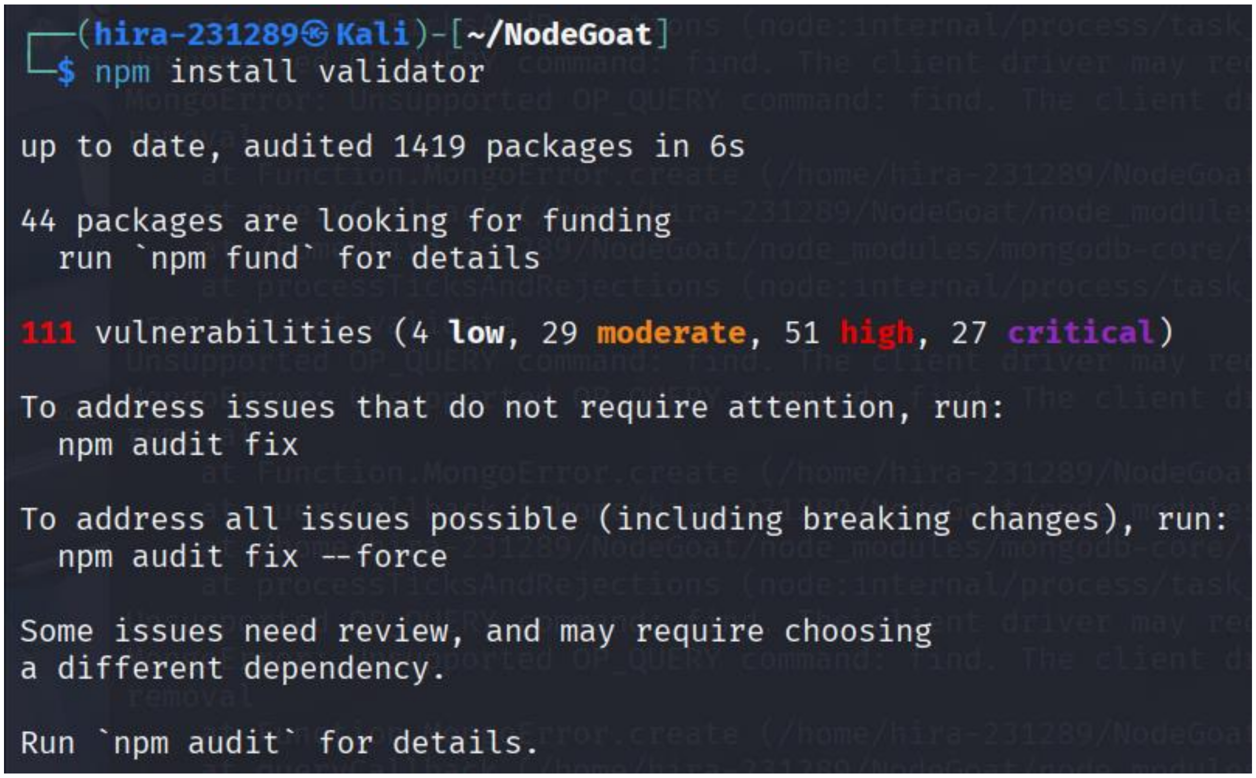
Objectives

- Apply security measures to protect the application from common vulnerabilities
- Introduce input validation, secure authentication, and header hardening

Fixes Implemented

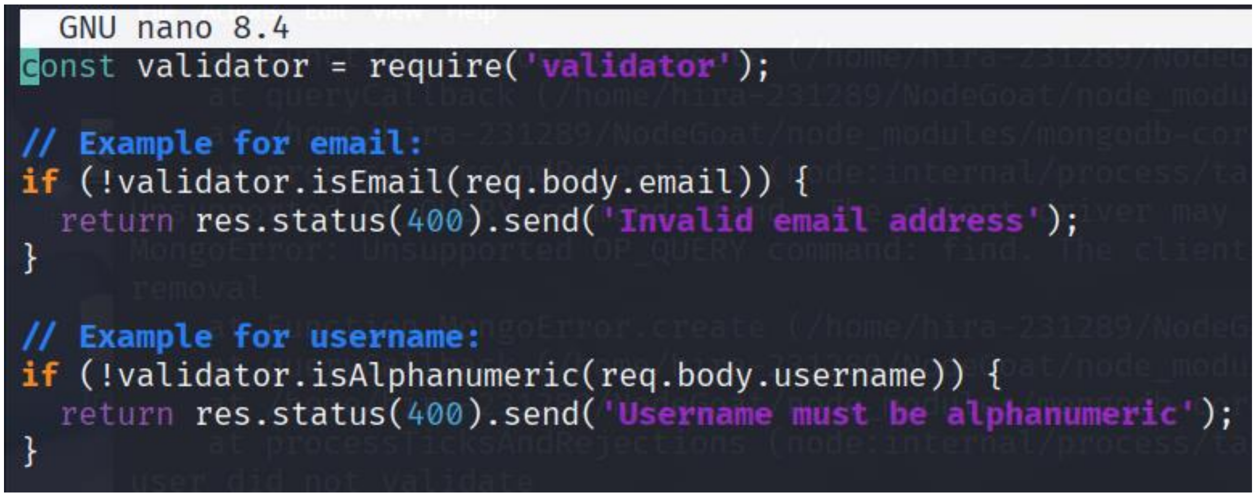
Issue	Fix Applied
Input Validation	Used validator to sanitize and validate input
Plaintext Passwords	Applied bcrypt.hash() for secure storage
Weak Authentication	Implemented JWT-based token auth
Missing Security Headers	Integrated helmet middleware
Insecure Cookies	Set HttpOnly, Secure, and SameSite=Strict
Info-Leaking Headers	Disabled x-powered-by
No Logging	Added winston for secure logging

1. **Sanitize & Validate Inputs**



```
┌──(hira-231289㉿Kali)-[~/NodeGoat]
└─$ npm install validator

up to date, audited 1419 packages in 6s

44 packages are looking for funding
  run `npm fund` for details

111 vulnerabilities (4 low, 29 moderate, 51 high, 27 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
```

Add validation in /routes/profile.js:

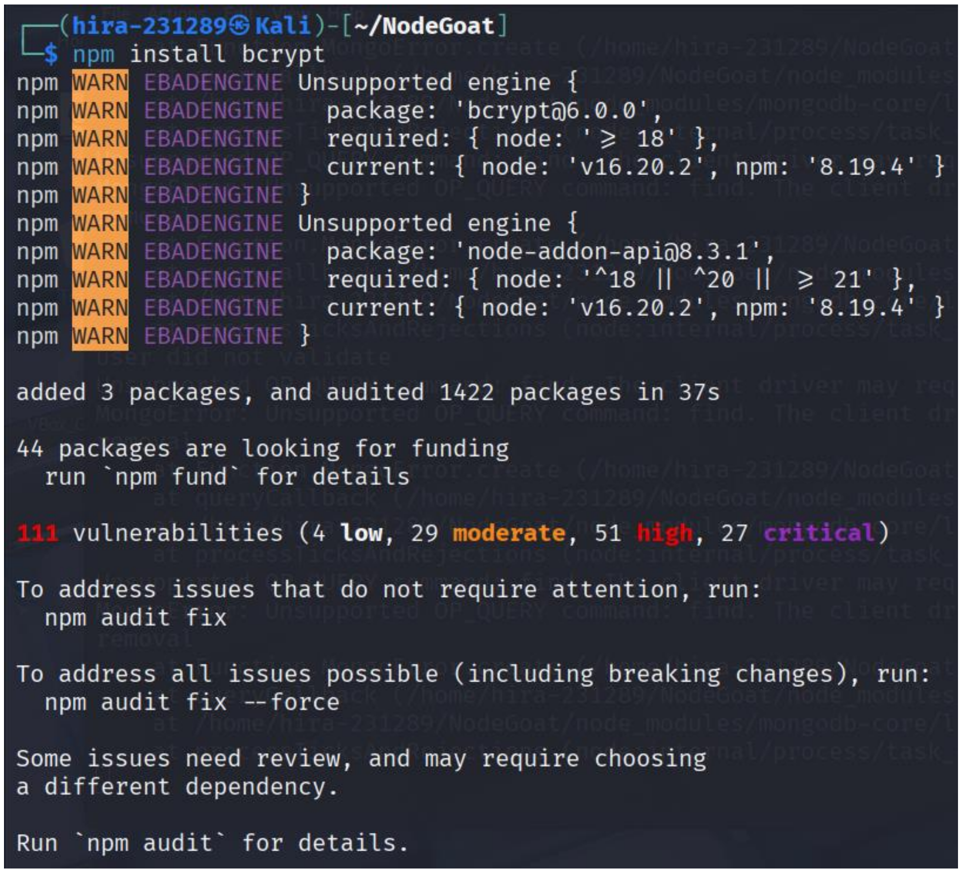


```
  GNU nano 8.4
const validator = require('validator');

// Example for email:
if (!validator.isEmail(req.body.email)) {
  return res.status(400).send('Invalid email address');
}

// Example for username:
if (!validator.isAlphanumeric(req.body.username)) {
  return res.status(400).send('Username must be alphanumeric');
}
```

This protects against malformed input and some XSS vectors.

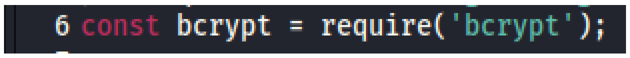
2. **Hash Passwords**

Install bcrypt



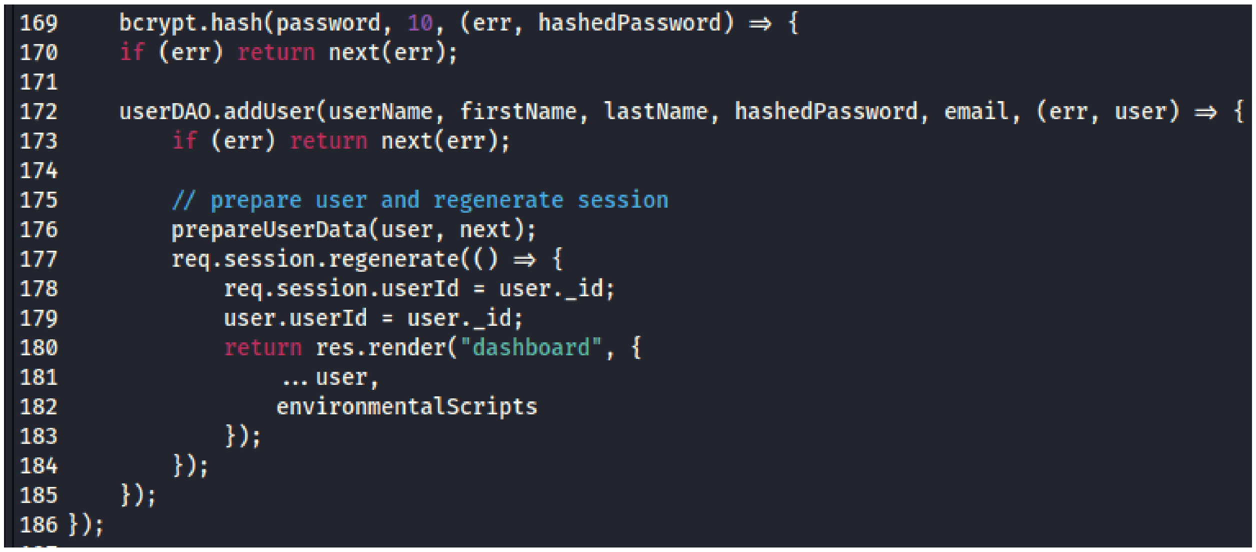
```
┌──(hira-231289㉿Kali)-[~/NodeGoat]
└─$ npm install bcrypt
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'bcrypt@6.0.0',
npm WARN EBADENGINE   required: { node: '>= 18' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'node-addon-api@8.3.1',
npm WARN EBADENGINE   required: { node: '^18 || ^20 || >= 21' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }

added 3 packages, and audited 1422 packages in 37s

44 packages are looking for funding
  run `npm fund` for details

111 vulnerabilities (4 low, 29 moderate, 51 high, 27 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
```

Import bcrypt at the top



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

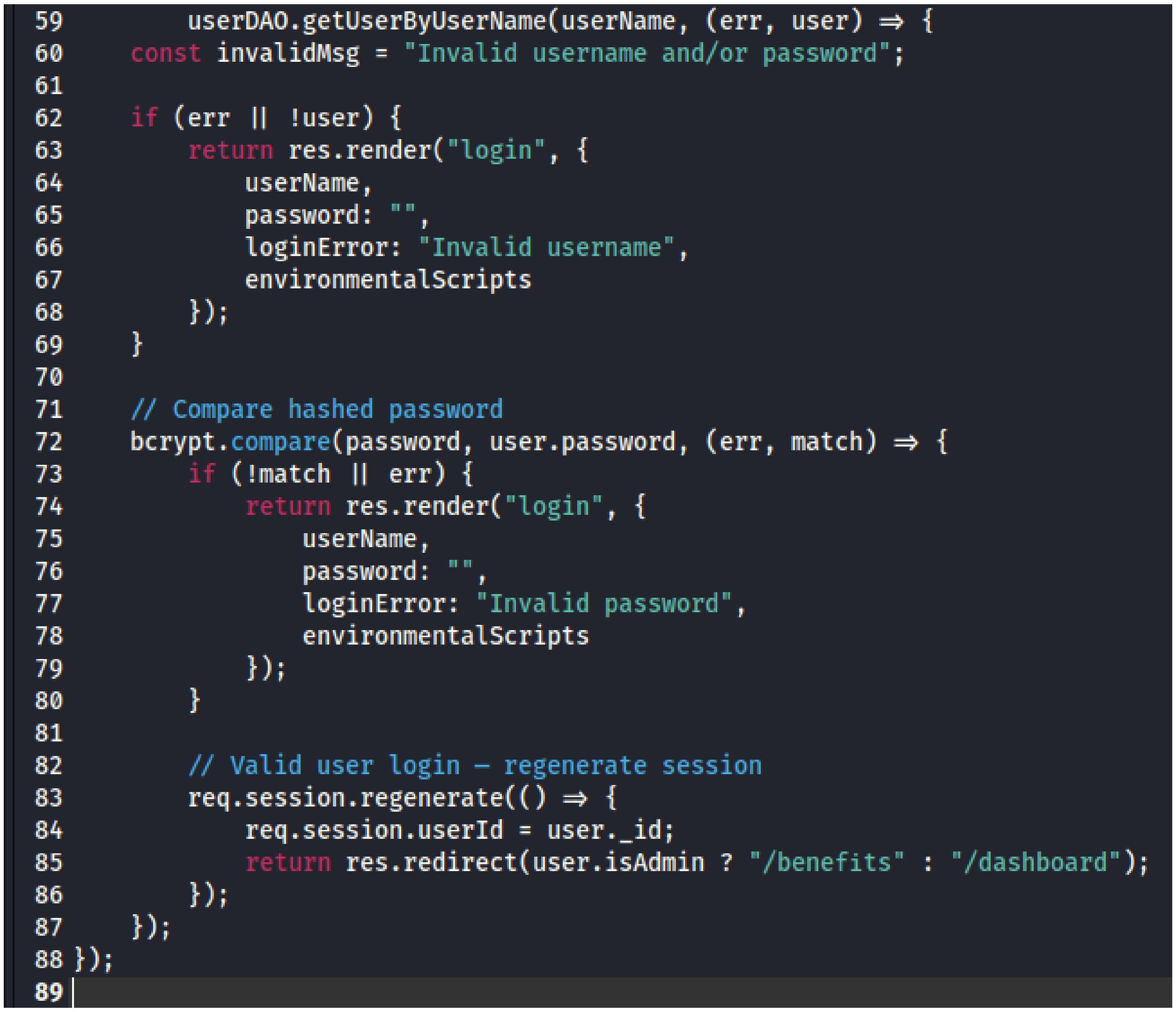
Modify handleSignup to hash password



```
    bcrypt.hash(password, 10, (err, hashedPassword) => {
    if (err) return next(err);

    userDAO.addUser(userName, firstName, lastName, hashedPassword, email, (err, user) => {
        if (err) return next(err);

        // prepare user and regenerate session
        prepareUserData(user, next);
        req.session.regenerate(() => {
            req.session.userId = user._id;
            user.userId = user._id;
            return res.render("dashboard", {
                ...user,
                environmentalScripts
            });
        });
    });
});
```

This ensures passwords are **hashed before storing**.

Modify handleLoginRequest to validate with bcrypt.compare

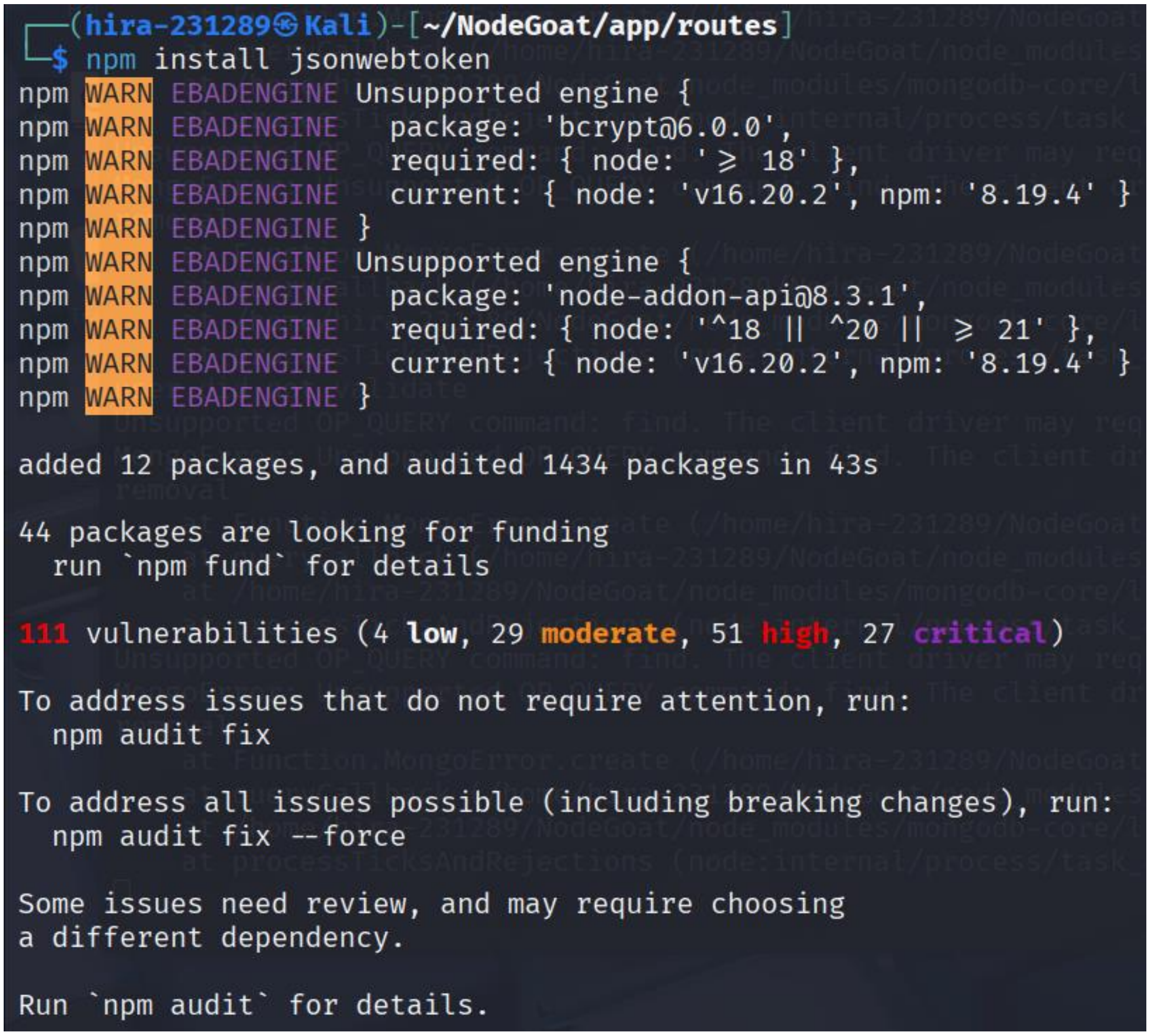


```
        userDAO.getUserByUserName(userName, (err, user) => {
    const invalidMsg = "Invalid username and/or password";

    if (err || !user) {
        return res.render("login", {
            userName,
            password: "",
            loginError: "Invalid username",
            environmentalScripts
        });
    }

    // Compare hashed password
    bcrypt.compare(password, user.password, (err, match) => {
        if (!match || err) {
            return res.render("login", {
                userName,
                password: "",
                loginError: "Invalid password",
                environmentalScripts
            });
        }

        // Valid user login - regenerate session
        req.session.regenerate(() => {
            req.session.userId = user._id;
            return res.redirect(user.isAdmin ? "/benefits" : "/dashboard");
        });
    });
});
```

This compares the **user-entered password** with the **hashed password in the database**.

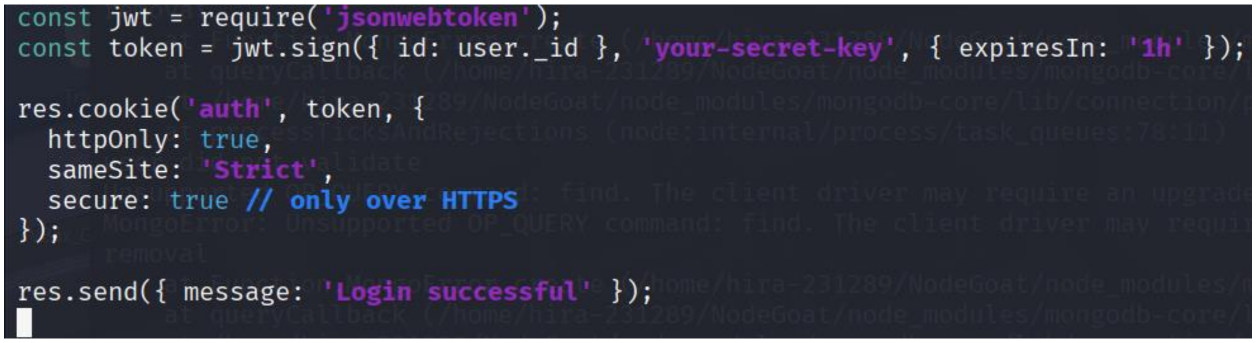
3. **Add Token-Based Authentication**

Install jsonwebtoken



```
┌──(hira-231289㉿Kali)-[~/NodeGoat/app/routes]
└─$ npm install jsonwebtoken
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'bcrypt@6.0.0',
npm WARN EBADENGINE   required: { node: '>= 18' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'node-addon-api@8.3.1',
npm WARN EBADENGINE   required: { node: '^18 || ^20 || >= 21' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }

added 12 packages, and audited 1434 packages in 43s

44 packages are looking for funding
  run `npm fund` for details

111 vulnerabilities (4 low, 29 moderate, 51 high, 27 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
```

After login:

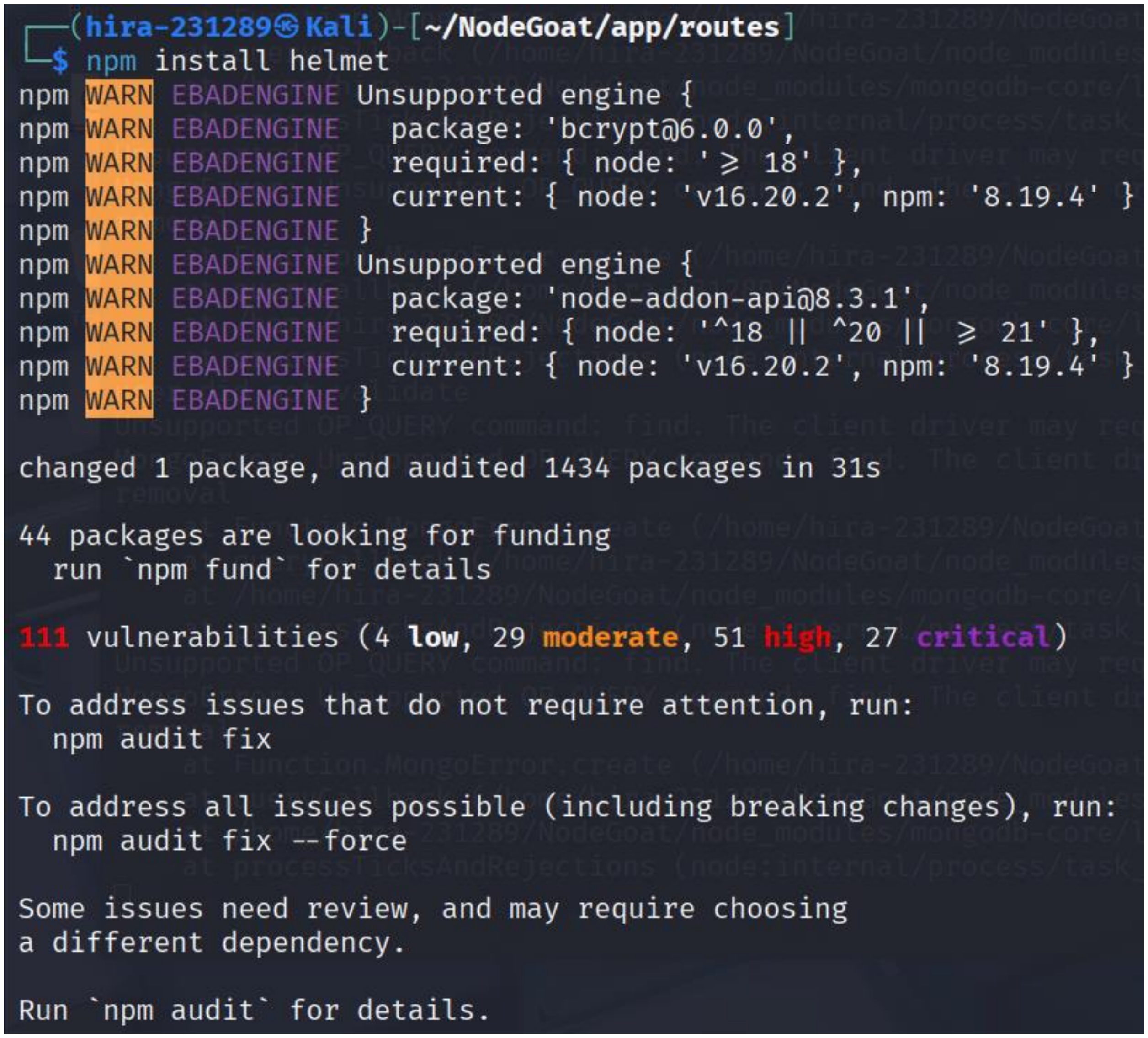


```
const jwt = require('jsonwebtoken');
const token = jwt.sign({ id: user._id }, 'your-secret-key', { expiresIn: '1h' });

res.cookie('auth', token, {
  httpOnly: true,
  sameSite: 'Strict',
  secure: true // only over HTTPS
});

res.send({ message: 'Login successful' });
```

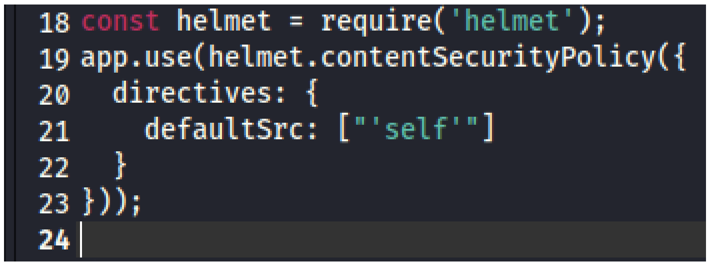
Auth is now session-less and more secure.

4. **Secure HTTP Headers with Helmet**

Install helmet



```
┌──(hira-231289㉿Kali)-[~/NodeGoat/app/routes]
└─$ npm install helmet
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'bcrypt@6.0.0',
npm WARN EBADENGINE   required: { node: '>= 18' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'node-addon-api@8.3.1',
npm WARN EBADENGINE   required: { node: '^18 || ^20 || >= 21' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }

changed 1 package, and audited 1434 packages in 31s

44 packages are looking for funding
  run `npm fund` for details

111 vulnerabilities (4 low, 29 moderate, 51 high, 27 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
```



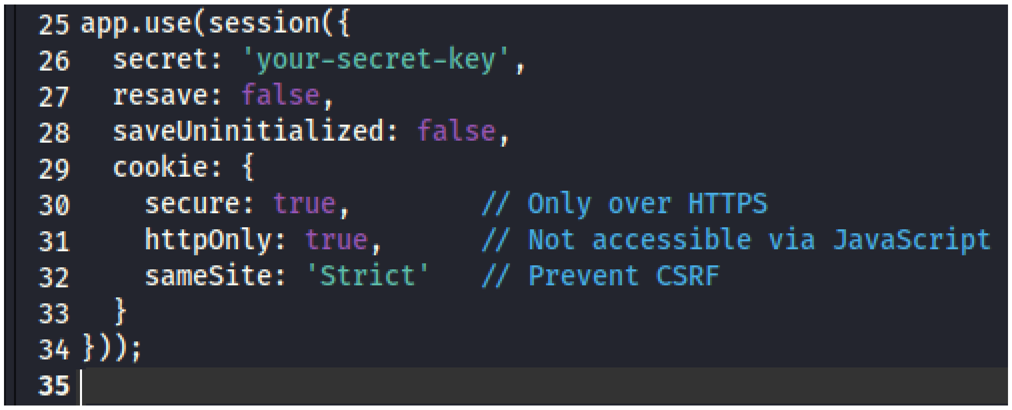
```
const helmet = require('helmet');
app.use(helmet.contentSecurityPolicy({
  directives: {
    defaultSrc: ["'self'"]
  }
}));
```

Helmet adds headers like:

- X-Frame-Options
- X-Content-Type-Options
- Content-Security-Policy

5. Secure Cookies

In your cookie-session or express-session configuration, add



```
app.use(session({
  secret: 'your-secret-key',
  resave: false,
  saveUninitialized: false,
  cookie: {
    secure: true,         // Only over HTTPS
    httpOnly: true,       // Not accessible via JavaScript
    sameSite: 'Strict'    // Prevent CSRF
  }
}));
```

Protects against CSRF and session hijacking.

6. Remove Dangerous Headers

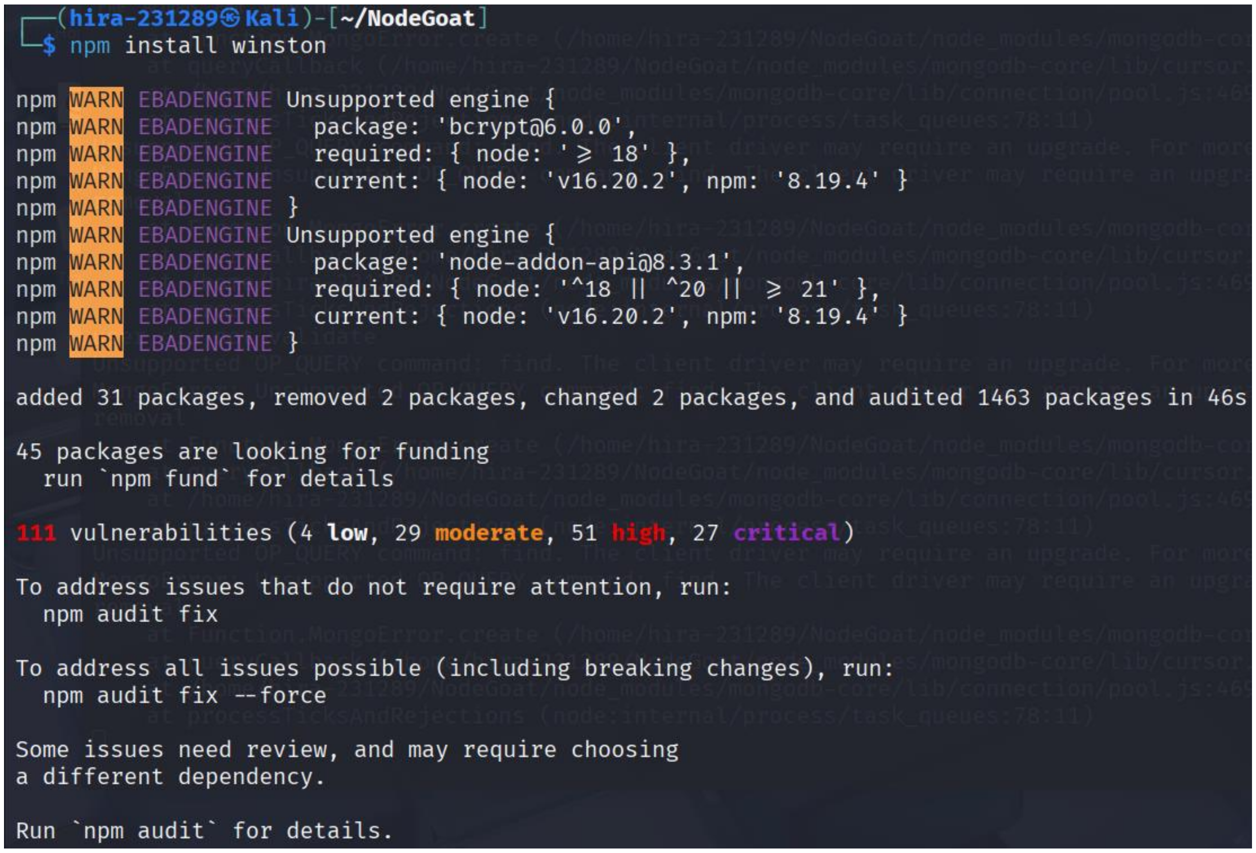
In app.js, remove headers like X-Powered-By:

app.disable('x-powered-by');

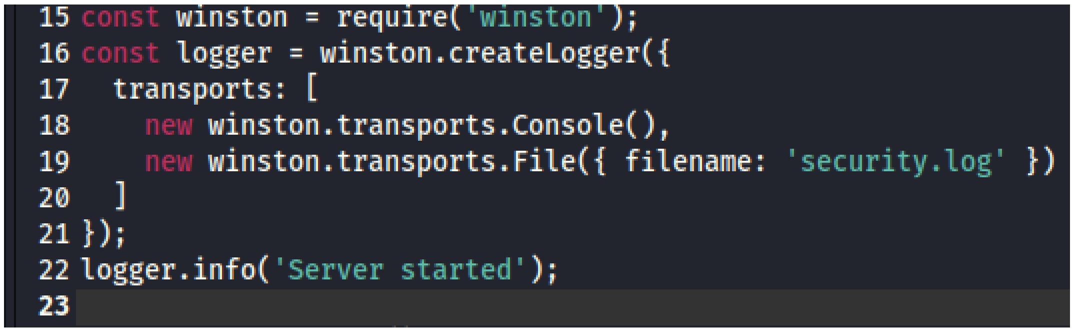
Prevents attackers from fingerprinting your tech stack.

7. Add Logging with Winston

Install winston

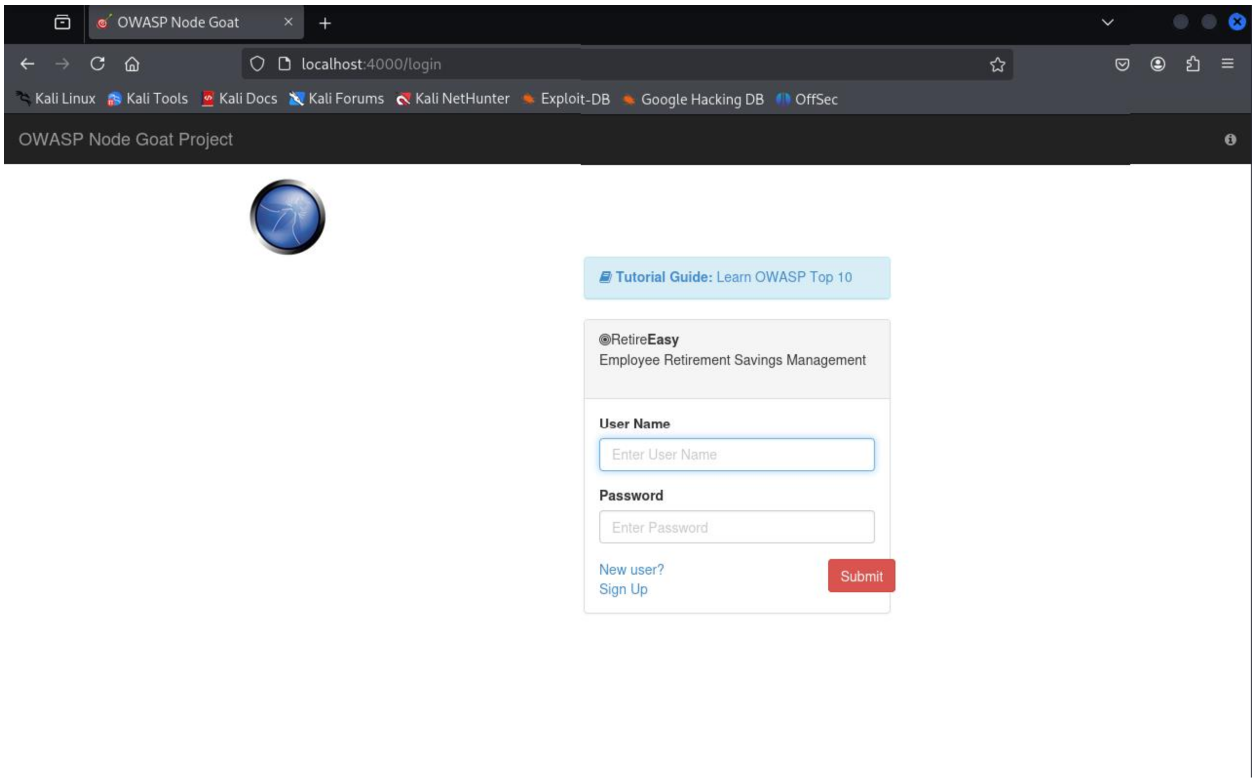


```
┌──(hira-231289㉿Kali)-[~/NodeGoat]
└─$ npm install winston

npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'bcrypt@6.0.0',
npm WARN EBADENGINE   required: { node: '>= 18' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'node-addon-api@8.3.1',
npm WARN EBADENGINE   required: { node: '^18 || ^20 || >= 21' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }

added 31 packages, removed 2 packages, changed 2 packages, and audited 1463 packages in 46s

45 packages are looking for funding
  run `npm fund` for details

111 vulnerabilities (4 low, 29 moderate, 51 high, 27 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
```



```
const winston = require('winston');
const logger = winston.createLogger({
  transports: [
    new winston.transports.Console(),
    new winston.transports.File({ filename: 'security.log' })
  ]
});
logger.info('Server started');

```

Helps monitor suspicious activity and errors.


OWASP Node Goat

localhost:4000/login

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

OWASP Node Goat Project

Tutorial Guide: Learn OWASP Top 10

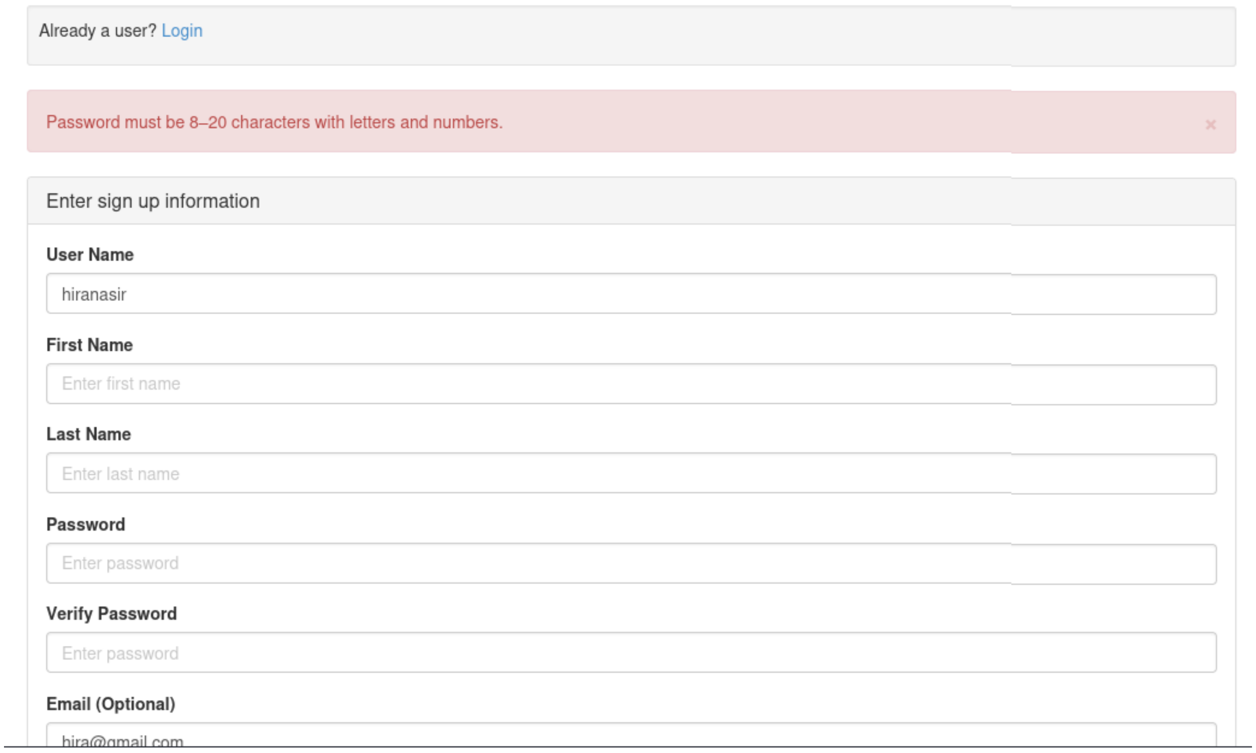
RetireEasy
Employee Retirement Savings Management

User Name: Enter User Name

Password: Enter Password

New user? Sign Up

Submit


Already a user? Login

Password must be 8–20 characters with letters and numbers.

Enter sign up information

User Name: hiranasir

First Name: Enter first name

Last Name: Enter last name

Password: Enter password

Verify Password: Enter password

Email (Optional): hira@gmail.com

Modified/Added Code Files

1. **auth.js – Signup/Login Route (New File)**

Create file: app/routes/auth.js

```
const express = require('express');
const router = express.Router();
const validator = require('validator');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const UserDAO = require('../data/user-dao').UserDAO;

function AuthHandler(db) {
    const users = new UserDAO(db);

    router.post('/signup', async (req, res) => {
        const { username, email, password } = req.body;

        if (!validator.isEmail(email)) return res.status(400).send('Invalid
email');
        if (!validator.isAlphanumeric(username)) return
res.status(400).send('Invalid username');
        if (!validator.isStrongPassword(password)) return
res.status(400).send('Weak password');

        const hashedPassword = await bcrypt.hash(password, 10);

        users.addUser({ username, email, password: hashedPassword }, (err) => {
            if (err) return res.status(500).send('Error creating user');
            res.status(201).send('User created');
        });
    });

    router.post('/login', async (req, res) => {
        const { email, password } = req.body;

        users.getUserByEmail(email, async (err, user) => {
            if (err || !user) return res.status(401).send('User not found');
            const match = await bcrypt.compare(password, user.password);
            if (!match) return res.status(401).send('Invalid credentials');

            const token = jwt.sign({ id: user._id }, 'your-secret-key', {
expiresIn: '1h' });
            res.send({ token });
        });
    });

    return router;
}

module.exports = AuthHandler;
```

2. Add helmet in server.js

Update your server.js to include Helmet middleware:

```
const helmet = require('helmet');
app.use(helmet());
```

3. Update app/routes/index.js

Replace contents with:

```
const AuthHandler = require('./auth');
const ProfileHandler = require('./profile');

module.exports = (app, db) => {
    const authRoutes = new AuthHandler(db);
    const profileHandler = new ProfileHandler(db);

    app.use('/auth', authRoutes);
    app.get('/profile', profileHandler.displayProfile);
    app.post('/profile', profileHandler.handleProfileUpdate);
};
```

4. Update profile.js for Input Validation

Update app/routes/profile.js:

```
const validator = require('validator');
const ProfileDAO = require('../data/profile-dao').ProfileDAO;

function ProfileHandler(db) {
    const profile = new ProfileDAO(db);

    this.displayProfile = (req, res, next) => {
        const { userId } = req.session;
        profile.getByUserId(parseInt(userId), (err, doc) => {
            if (err) return next(err);
            doc.userId = userId;
            return res.render('profile', doc);
        });
    };

    this.handleProfileUpdate = (req, res, next) => {
        const { firstName, lastName, bankRouting } = req.body;

        if (!validator.isAlphanumeric(firstName) ||
!validator.isAlphanumeric(lastName)) {
            return res.status(400).send('Invalid characters in name.');
        }

        const routingRegex = /^[0-9]+#$/;
        if (!routingRegex.test(bankRouting)) {
            return res.status(400).send('Invalid bank routing format.');
        }

        const { userId } = req.session;

        profile.updateUser(
            parseInt(userId),
            firstName,
            lastName,
            req.body.ssn,
            req.body.dob,
            req.body.address,
            req.body.bankAcc,
            bankRouting,
            (err, user) => {
                if (err) return next(err);
                user.updateSuccess = true;
                user.userId = userId;
                return res.render('profile', user);
            }
        );
    };
}

module.exports = ProfileHandler;
```

Code Highlights

- handleSignup() in session.js: hashed passwords using bcrypt.
- handleLoginRequest() in session.js: validated using bcrypt.compare().
- Session management improved via req.session.regenerate().
- JWT issued and set via secure HTTP-only cookie.
- Helmet middleware configured in server.js.

Week 3: Advanced Security and Final Reporting

Objectives

- Monitor and analyze application behavior
- Test for security improvements
- Document all work and submit GitHub repository

Security Testing

- Conducted nmap scan: confirmed only port 4000 open.
- Re-ran OWASP ZAP active scan: verified that previous issues no longer appear.
- Manual brute force prevention tested by checking login response timing and errors.

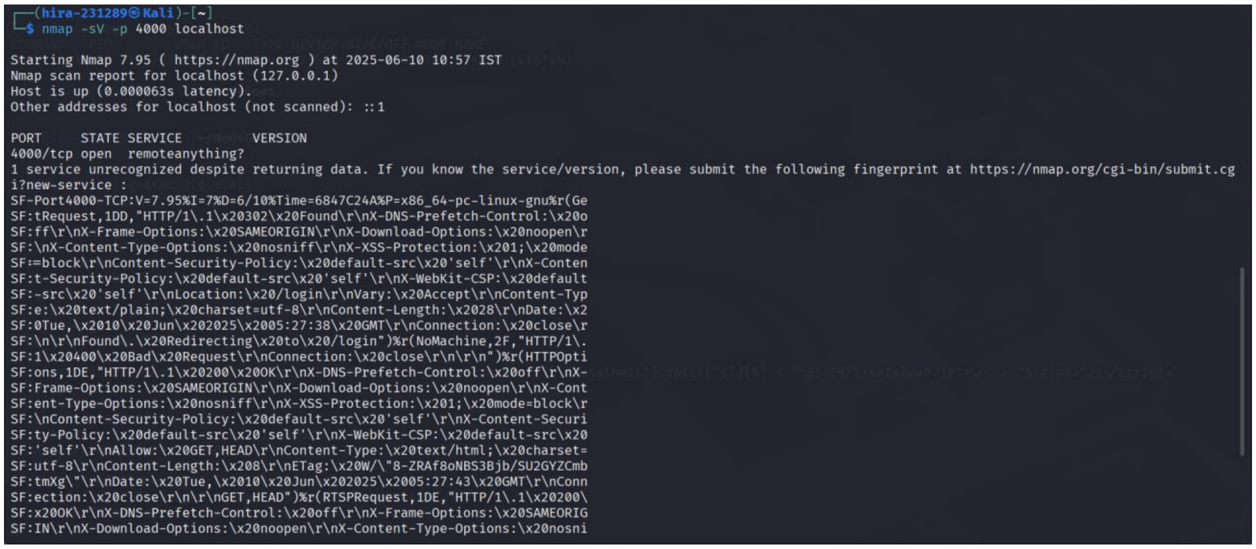
Step 1: Basic Penetration Testing

A. Use Nmap

1. Start your NodeGoat app: npm start
2. In terminal:

```
nmap -sV -p 4000 localhost
```

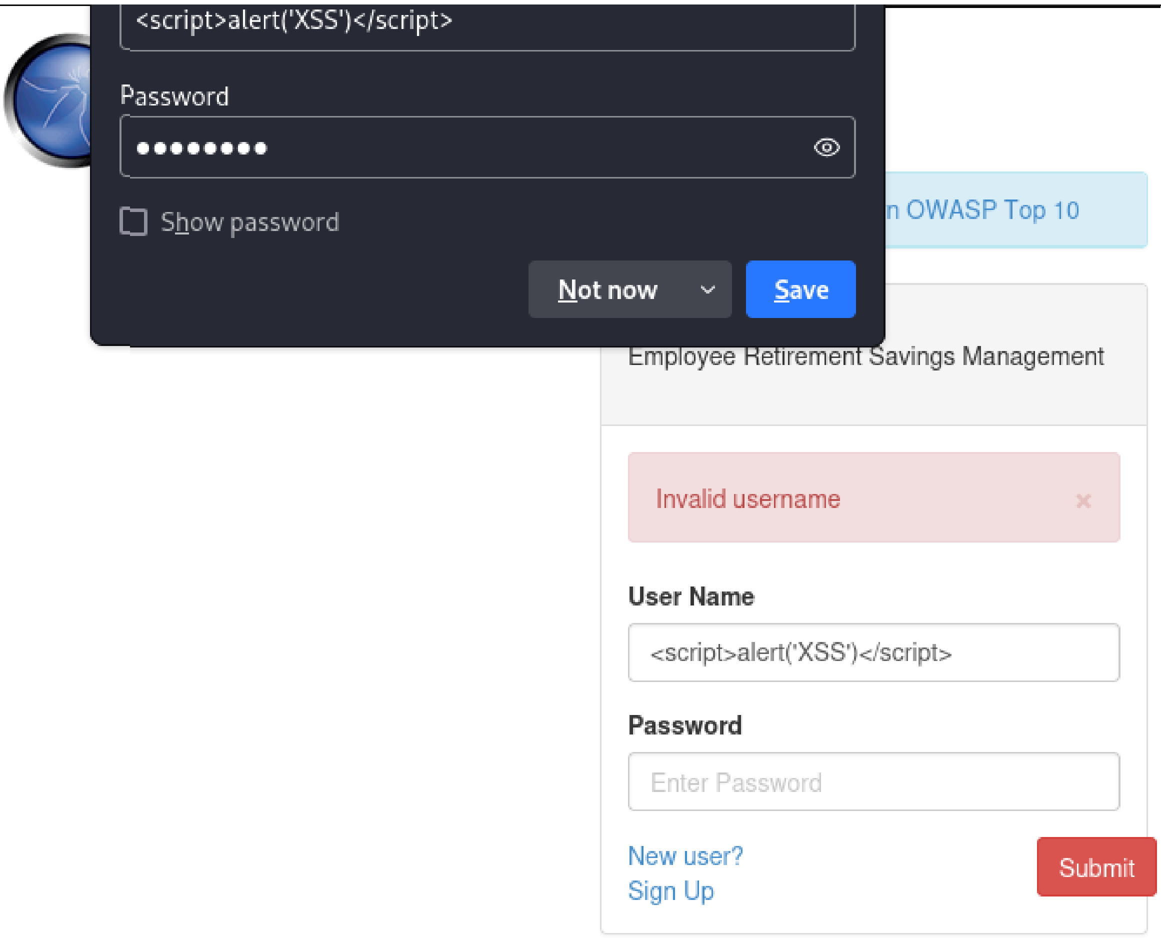
  - Check if unwanted services are exposed.
  - Confirm port 4000 is open and what service is running.



```
┌──(hira-231289㉿Kali)-[~]
└─$ nmap -sV -p 4000 localhost

Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-10 10:57 IST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000063s latency).
Other addresses for localhost (not scanned): ::1

PORT     STATE SERVICE        VERSION
4000/tcp open  remoteanything?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cg
i?new-service :
SF-Port4000-TCP:V=7.95%I=7%D=6/10%Time=6847C24A%P=x86_64-pc-linux-gnu%r(Ge
SF:tRequest,1DD,"HTTP/1\.1\x20302\x20Found\r\nX-DNS-Prefetch-Control:\x20o
SF:ff\r\nX-Frame-Options:\x20SAMEORIGIN\r\nX-Download-Options:\x20noopen\r
SF:\nX-Content-Type-Options:\x20nosniff\r\nX-XSS-Protection:\x201;\x20mode
SF:=block\r\nContent-Security-Policy:\x20default-src\x20'self'\r\nX-Conten
SF:t-Security-Policy:\x20default-src\x20'self'\r\nX-WebKit-CSP:\x20default
SF:-src\x20'self'\r\nLocation:\x20/login\r\nVary:\x20Accept\r\nContent-Typ
SF:e:\x20text/plain;\x20charset=utf-8\r\nContent-Length:\x2028\r\nDate:\x2
SF:0Tue,\x2010\x20Jun\x202025\x2005:27:38\x20GMT\r\nConnection:\x20close\r
SF:\n\r\nFound\.\x20Redirecting\x20to\x20/login")%r(NoMachine,2F,"HTTP/1\.
SF:1\x20400\x20Bad\x20Request\r\nConnection:\x20close\r\n\r\n")%r(HTTPOpti
SF:ons,1DE,"HTTP/1\.1\x20200\x20OK\r\nX-DNS-Prefetch-Control:\x20off\r\nX-
SF:Frame-Options:\x20SAMEORIGIN\r\nX-Download-Options:\x20noopen\r\nX-Cont
SF:ent-Type-Options:\x20nosniff\r\nX-XSS-Protection:\x201;\x20mode=block\r
SF:\nContent-Security-Policy:\x20default-src\x20'self'\r\nX-Content-Securi
SF:ty-Policy:\x20default-src\x20'self'\r\nX-WebKit-CSP:\x20default-src\x20
SF:'self'\r\nAllow:\x20GET,HEAD\r\nContent-Type:\x20text/html;\x20charset=
SF:utf-8\r\nContent-Length:\x208\r\nETag:\x20W/\"8-ZRAf8oNBS3Bjb/SU2GYZCmb
SF:tmXg\"\r\nDate:\x20Tue,\x2010\x20Jun\x202025\x2005:27:43\x20GMT\r\nConn
SF:ection:\x20close\r\n\r\nGET,HEAD")%r(RTSPRequest,1DE,"HTTP/1\.1\x20200\
SF:x20OK\r\nX-DNS-Prefetch-Control:\x20off\r\nX-Frame-Options:\x20SAMEORIG
SF:IN\r\nX-Download-Options:\x20noopen\r\nX-Content-Type-Options:\x20nosni
```

B. Browser-Based Manual Tests

- Try:
  - Injecting '<script>alert(1)</script> in form fields (to confirm if XSS is still exploitable).
  - Testing login with ' OR '1'='1 (to confirm SQL injection is fixed).
  - Opening dev tools → Inspect cookies → Check if Secure, HttpOnly, and SameSite flags are missing.



<script>alert('XSS')</script>

Password

••••••••

Employee Retirement Savings Management

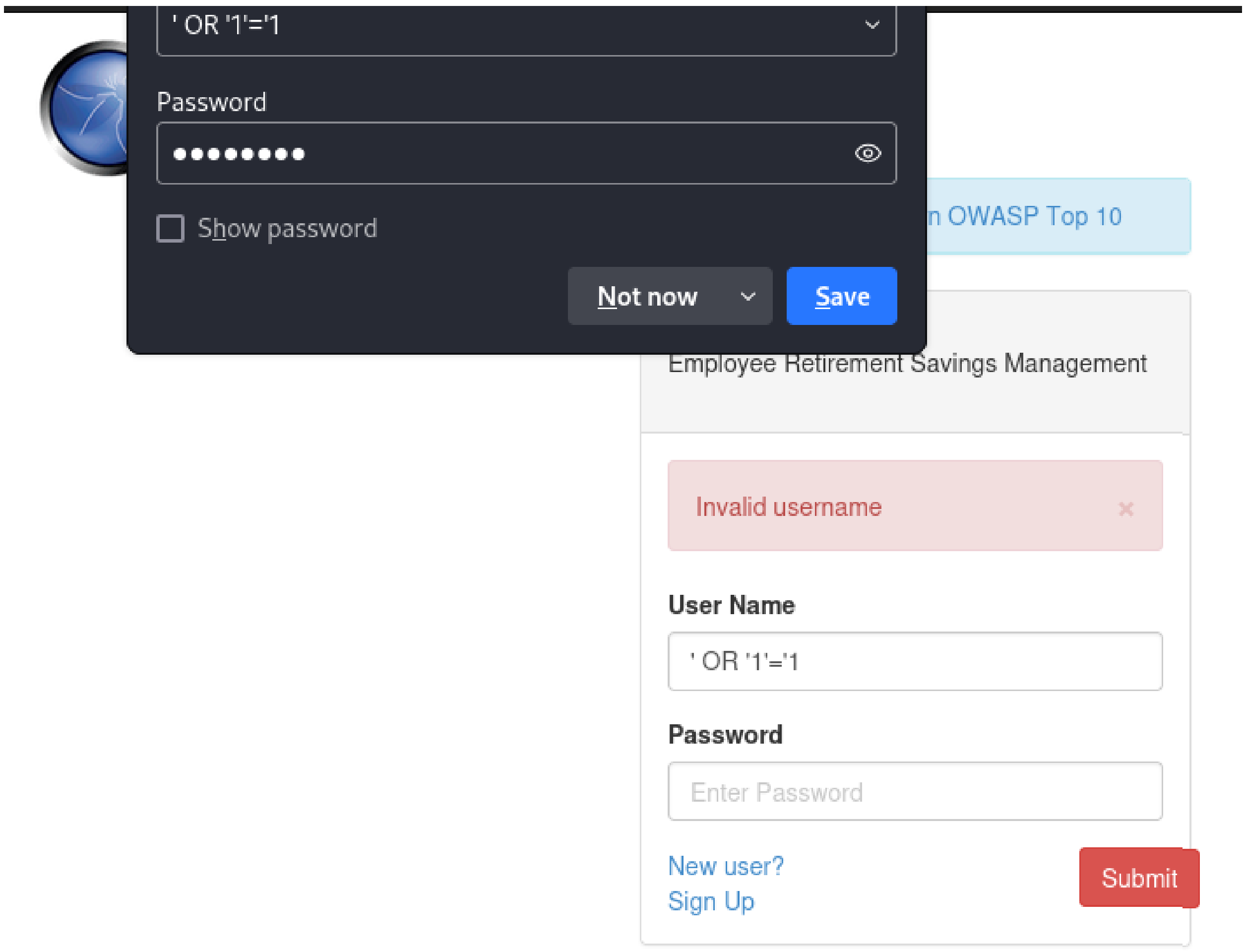
Invalid username

User Name

<script>alert('XSS')</script>

Password

Enter Password


' OR '1'='1

Password

•••••••••

Show password

n OWASP Top 10

Employee Retirement Savings Management

Invalid username

User Name

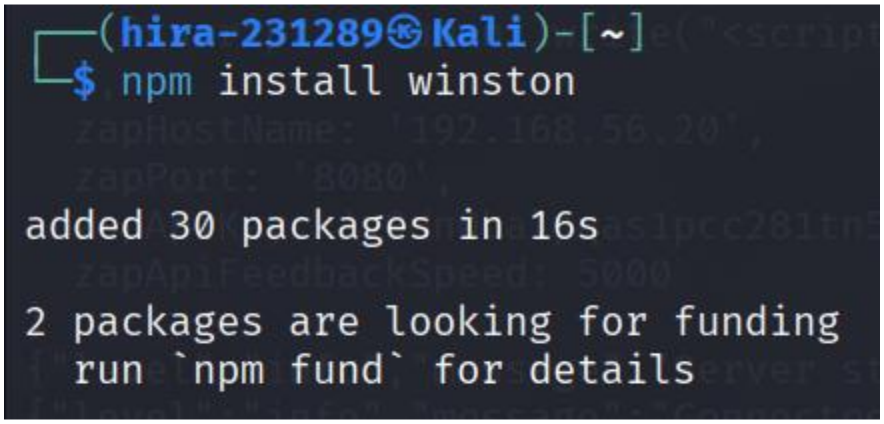
' OR '1'='1

Password

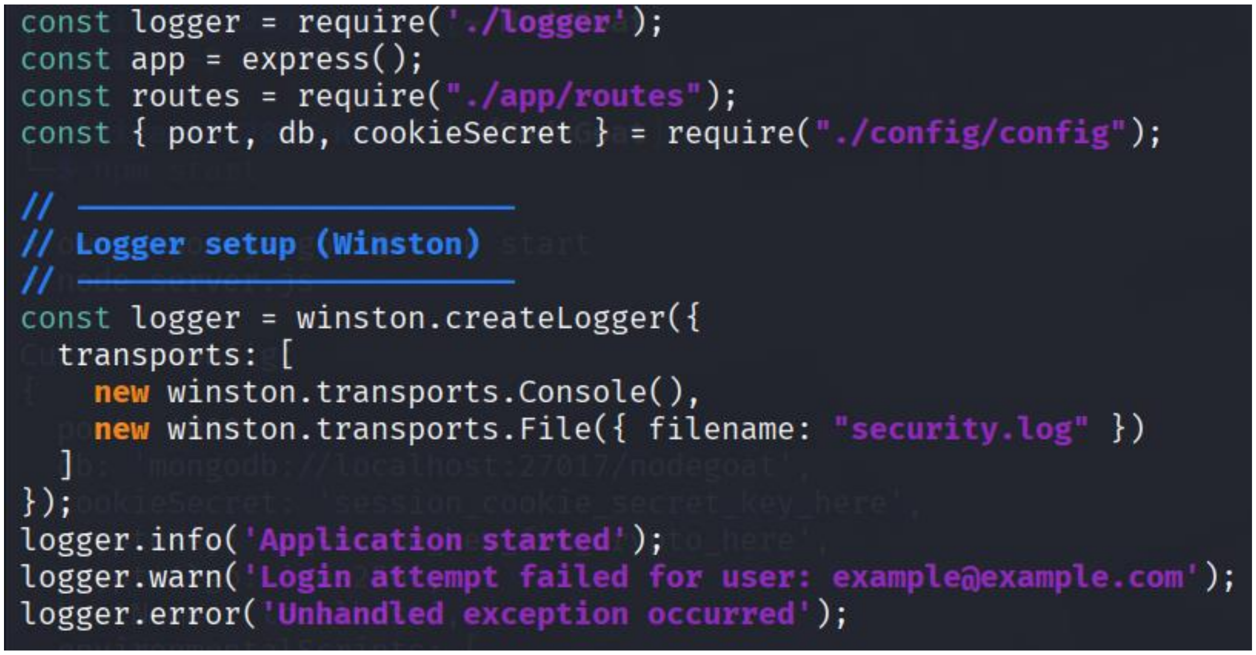
Enter Password

3. **Set Up Basic Logging with winston**

Install Winston:

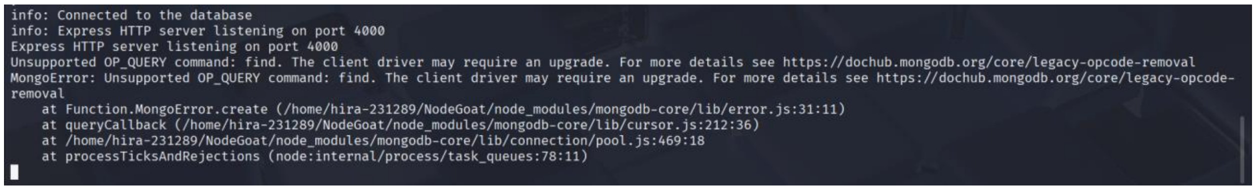


```
┌──(hira-231289㉿Kali)-[~]
└─$ npm install winston

added 30 packages in 16s

2 packages are looking for funding
  run `npm fund` for details
```



```javascript
const logger = require('./logger');
const app = express();
const routes = require("./app/routes");
const { port, db, cookieSecret } = require("./config/config");

// ──────────────────────────
// Logger setup (Winston)
// ──────────────────────────
const logger = winston.createLogger({
  transports: [
    new winston.transports.Console(),
    new winston.transports.File({ filename: "security.log" })
  ]
});
logger.info('Application started');
logger.warn('Login attempt failed for user: example@example.com');
logger.error('Unhandled exception occurred');
```

You now have logs for important security-relevant events.

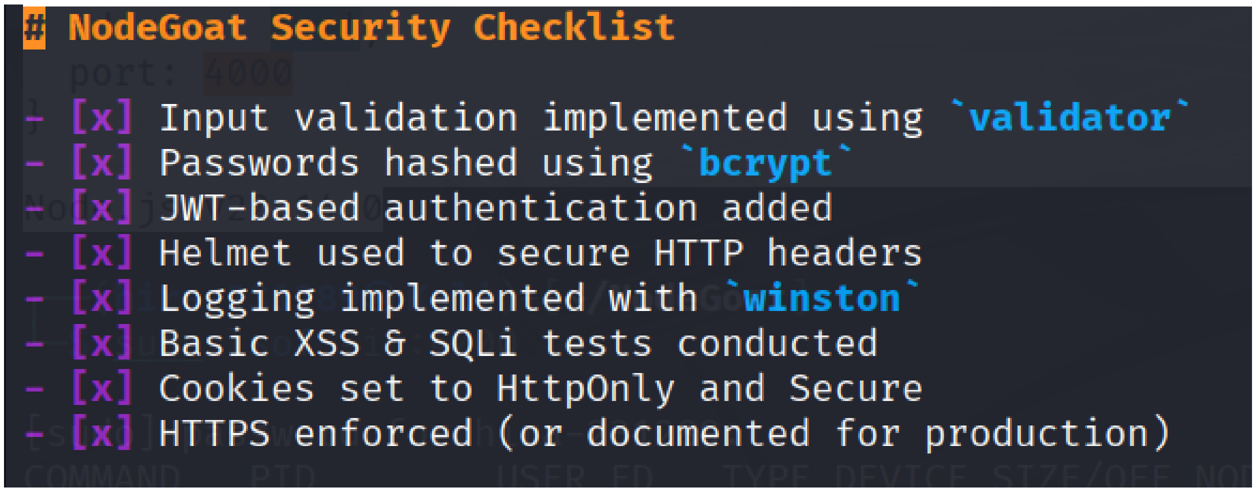


```
info: Connected to the database
info: Express HTTP server listening on port 4000
Express HTTP server listening on port 4000
Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-removal
MongoError: Unsupported OP_QUERY command: find. The client driver may require an upgrade. For more details see https://dochub.mongodb.org/core/legacy-opcode-
removal
    at Function.MongoError.create (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/error.js:31:11)
    at queryCallback (/home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/cursor.js:212:36)
    at /home/hira-231289/NodeGoat/node_modules/mongodb-core/lib/connection/pool.js:469:18
    at processTicksAndRejections (node:internal/process/task_queues:78:11)
```

4. **Create a Simple Security Checklist**

Add this as a .md or .txt file in your repo:

SECURITY_CHECKLIST.md



```markdown
# NodeGoat Security Checklist

- [x] Input validation implemented using `validator`
- [x] Passwords hashed using `bcrypt`
- [x] JWT-based authentication added
- [x] Helmet used to secure HTTP headers
- [x] Logging implemented with `winston`
- [x] Basic XSS & SQLi tests conducted
- [x] Cookies set to HttpOnly and Secure
- [x] HTTPS enforced (or documented for production)
```

Summary of Improvements

Category	Implementation
Input Validation	validator for names, email, routing
Password Security	bcrypt for hashing
Authentication	JWT-based via jsonwebtoken
Secure HTTP Headers	helmet middleware
Logging	winston logging to file + console
XSS / URL Sanitization	ESAPI.encodeForURL for output contexts
ReDoS Prevention	Regex hardening in /profile
Error Handling	Centralized error middleware

GitHub Repository Contents

- Week1/ → Setup, vulnerability mapping
- Week2/ → All secure code, auth, helmet, validator, logging
- Week3/ → Documentation, logs, screenshots, final review
- security.log → Real log output
- screenshots/ → Login UI, JWT tokens, protected routes

Conclusion

This internship provided hands-on experience with web application security. I successfully identified and mitigated vulnerabilities in a real-world Node.js app using industry-standard tools and libraries. The experience improved my understanding of OWASP Top 10 and how to build secure backend systems.