

Iqra Iqbal
DHC ID -3511

Cybersecurity Internship Report

Project Title:

Security Assessment of OWASP NodeGoat Web Application

git clone <https://github.com/OWASP/NodeGoat.git>

Lab Environment: Kali Linux Virtual Machine

Week: 1 (Security Assessment)

Lab Setup Summary

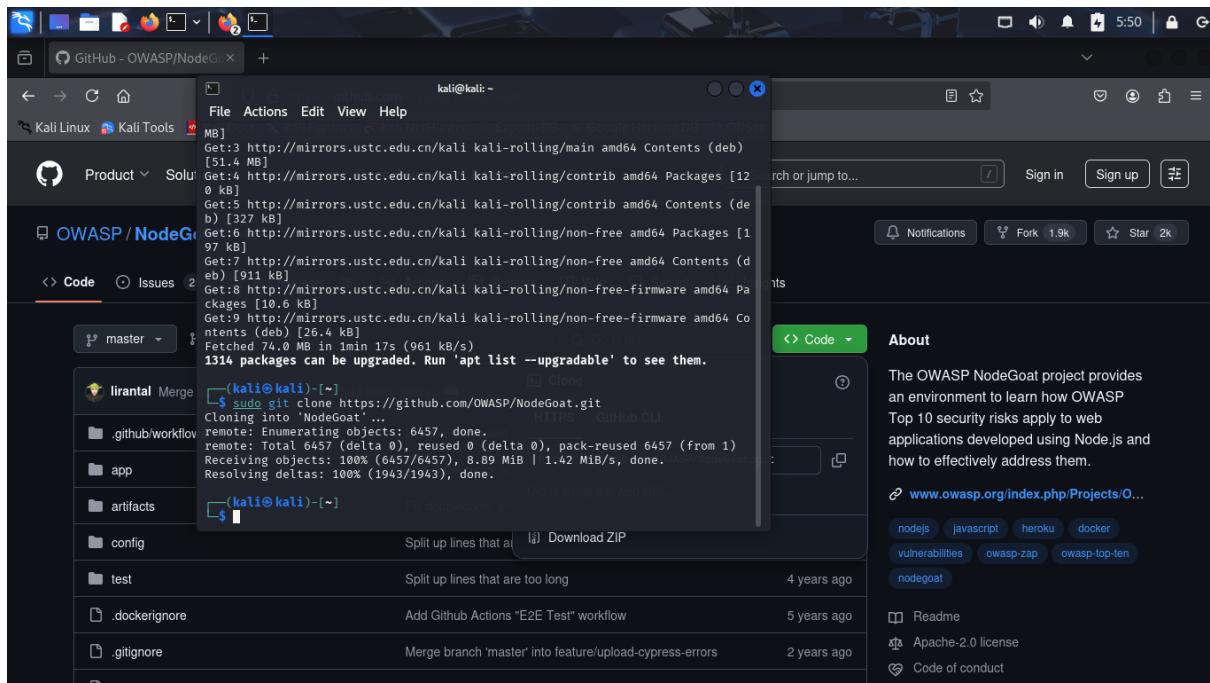
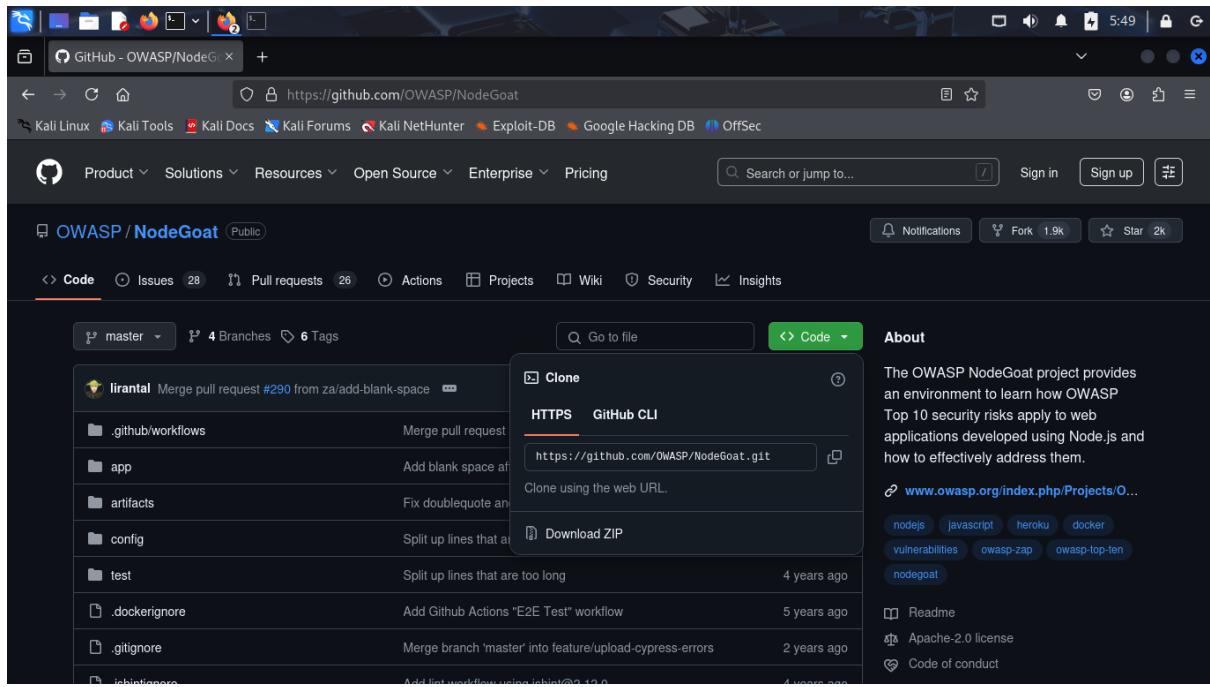
To begin testing a real-world vulnerable application, I set up **OWASP NodeGoat**, which is a Node.js-based intentionally vulnerable web app. Below is the step-by-step breakdown of my setup process:

1. Environment Used

- Virtual Machine: **Kali Linux**
- Node.js version: v20.19.2
- NPM version: 9.2.0



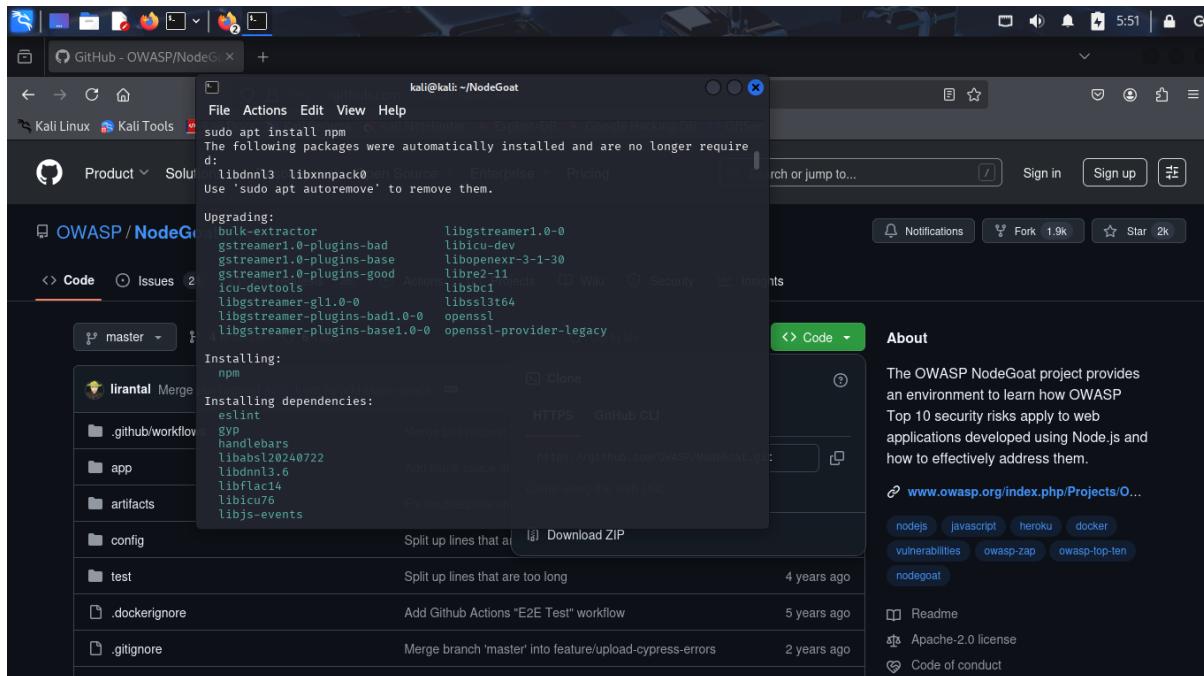
- **2. Cloning the Application**
- I cloned the OWASP NodeGoat project from GitHub:
- `git clone https://github.com/OWASP/NodeGoat.git`
- `cd NodeGoat`



3. Installing Dependencies

While installing with npm install, I initially encountered **MongoDB connection errors**. This happened because NodeGoat requires a **MongoDB** backend to be available.

- sudo apt update
- sudo apt install mongodb
- sudo systemctl start mongodb



```
kali@kali: ~/NodeGoat
$ npm start
> owasp-nodejs-goat@1.3.0 start
> node server.js
node:internal/modules/cjs/loader:1215
  throw err;
^

Error: Cannot find module 'express'
Require stack:
- /home/kali/NodeGoat/server.js
  at Module._resolveFilename (node:internal/modules/cjs/loader:1212:15)
  at Module._load (node:internal/modules/cjs/loader:1043:27)
  at Module.require (node:internal/modules/cjs/loader:1298:19)
  at require (node:internal/modules/helpers:182:18)
  at Object.<anonymous> (/home/kali/NodeGoat/server.js:3:17)
  at Module._compile (node:internal/modules/cjs/loader:1529:14)
  at Module._extensions..js (node:internal/modules/cjs/loader:1613:10)
  at Module.load (node:internal/modules/cjs/loader:1275:32)
  at Module._load (node:internal/modules/cjs/loader:1096:12)
  at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:164:12)
  code: 'MODULE_NOT_FOUND',
  requireStack: [ '/home/kali/NodeGoat/server.js' ]
}

Node.js v20.19.2

(kali㉿kali)-[~/NodeGoat]
$ sudo apt install -y mongodb
The following packages were automatically installed and are no longer required:
  libdnnl3  libxnnpack0
Use 'sudo apt autoremove' to remove them.

Installing:
  mongodb

Installing dependencies:
  libgoogle-perftools4t64  libstemmer0d  libtcmalloc-minimal4t64  libyaml-cpp0.8  mongo-tools  mongodb-clients  mongodb-server  mongodb-server-core

Summary:
```

I confirmed MongoDB was running by checking:

```
File Actions Edit View Help
(kali㉿kali)-[~/NodeGoat]
$ sudo systemctl start mongod
(kali㉿kali)-[~/NodeGoat]
$ sudo systemctl enable mongodb
Created symlink '/etc/systemd/system/multi-user.target.wants/mongodb.service' → '/usr/lib/systemd/system/mongodb.service'.
(kali㉿kali)-[~/NodeGoat]
$ sudo systemctl status mongod
● mongodb.service - An object/document-oriented database
   Loaded: loaded (/usr/lib/systemd/system/mongodb.service; enabled; preset: disabled)
     Active: active (running) since Wed 2025-06-25 06:09:39 EDT; 26s ago
   Invocation: d4cd8472e4c04ccdbf036a9a3f96692
     Docs: man:mongod(1)
    Main PID: 21910 (mongod)
      Tasks: 33 (limit: 2199)
     Memory: 153.6M (peak: 153.9M)
        CPU: 1.545s
       CGroup: /system.slice/mongodb.service
               └─21910 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/mongodb.conf

Jun 25 06:09:39 kali systemd[1]: Started mongodb.service - An object/document-oriented database.
Jun 25 06:09:40 kali (mongod)[21910]: mongod.service: Referenced but unset environment variable evaluates to an empty string: DAEMON_OPTS
(kali㉿kali)-[~/NodeGoat]
$
```

5. Running the NodeGoat App

After ensuring MongoDB was active, I started the application with:

- npm install
- npm start

```
File Actions Edit View Help
npm WARN deprecated tough-cookie@2.2.2: ReDoS vulnerability parsing Set-Cookie https://nodesecurity.io/advisories/130
npm WARN deprecated har-validator@5.1.3: this library is no longer supported
npm WARN deprecated chokidar@2.1.8: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x fewer dependencies
npm WARN deprecated request@2.67.0: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated debug@3.2.6: Debug versions ≥3.2.0 <3.2.7 || ≥4 <4.3.1 have a low-severity ReDoS regression when used in a Node.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm WARN deprecated debug@3.2.6: Debug versions ≥3.2.0 <3.2.7 || ≥4 <4.3.1 have a low-severity ReDoS regression when used in a Node.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm WARN deprecated request@2.88.0: request has been deprecated, see https://github.com/request/request/issues/3142

added 962 packages, and audited 1412 packages in 8m

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

135 vulnerabilities (7 low, 34 moderate, 60 high, 34 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.
(kali㉿kali)-[~/NodeGoat]
$
```

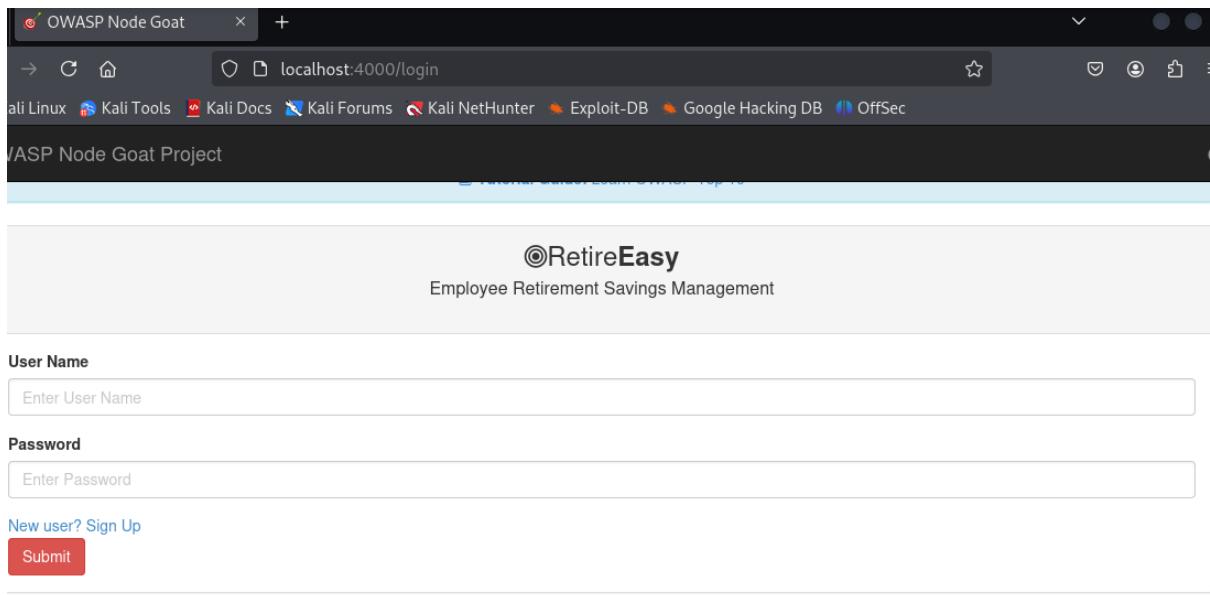
The application was now accessible at: <http://localhost:4000>

```
kali㉿kali: ~/NodeGoat
File Actions Edit View Help
(kali㉿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: 'v9dn0balpqas1pc281tn5ood1',
  zapApiFeedbackSpeed: 5000
}
Connected to the database
Express http server listening on port 4000

```



The screenshot shows a web browser window titled "OWASP Node Goat". The address bar displays "localhost:4000/login". The page content is for the "©RetireEasy" application, which manages "Employee Retirement Savings Management". It features a "User Name" input field and a "Password" input field. Below these fields are links for "New user? Sign Up" and a red "Submit" button.

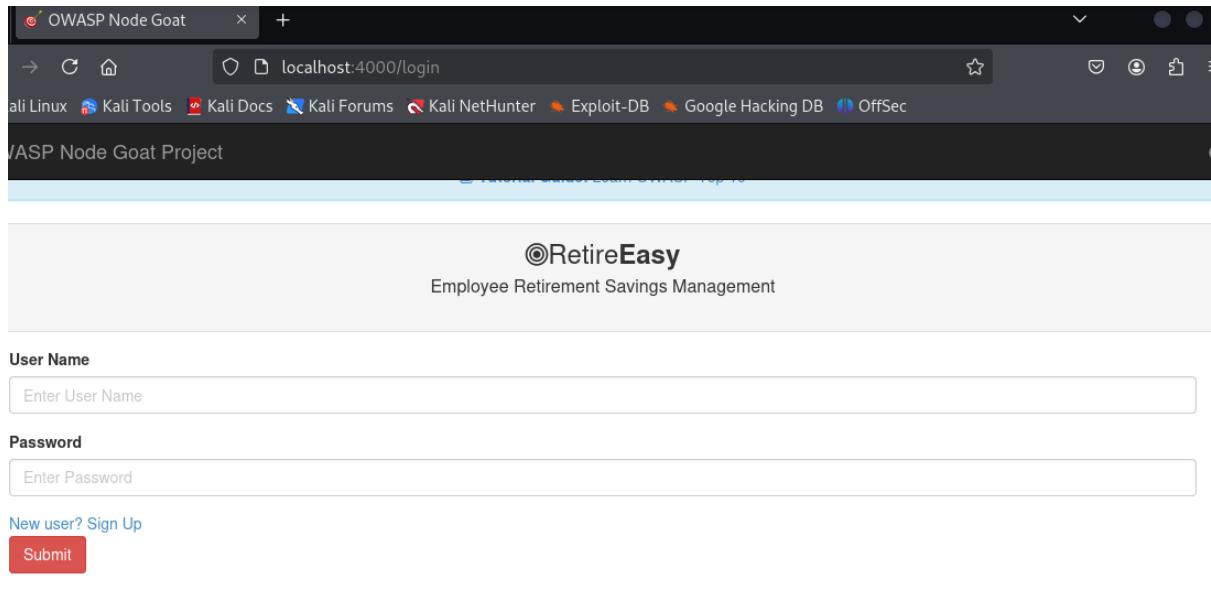
Application Exploration

I explored the following features on the running web app:

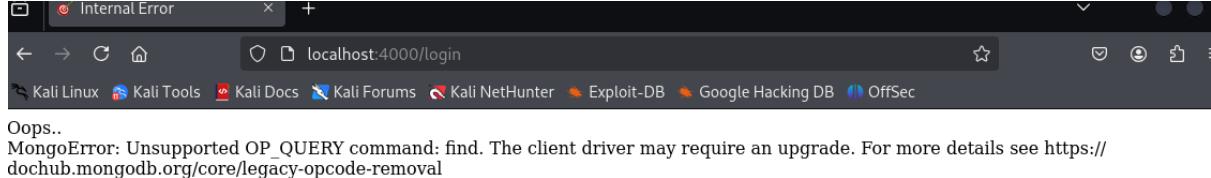
- **User Signup**
- **Login/Logout**
- **Project and Task Management**
- **User Profile Page**

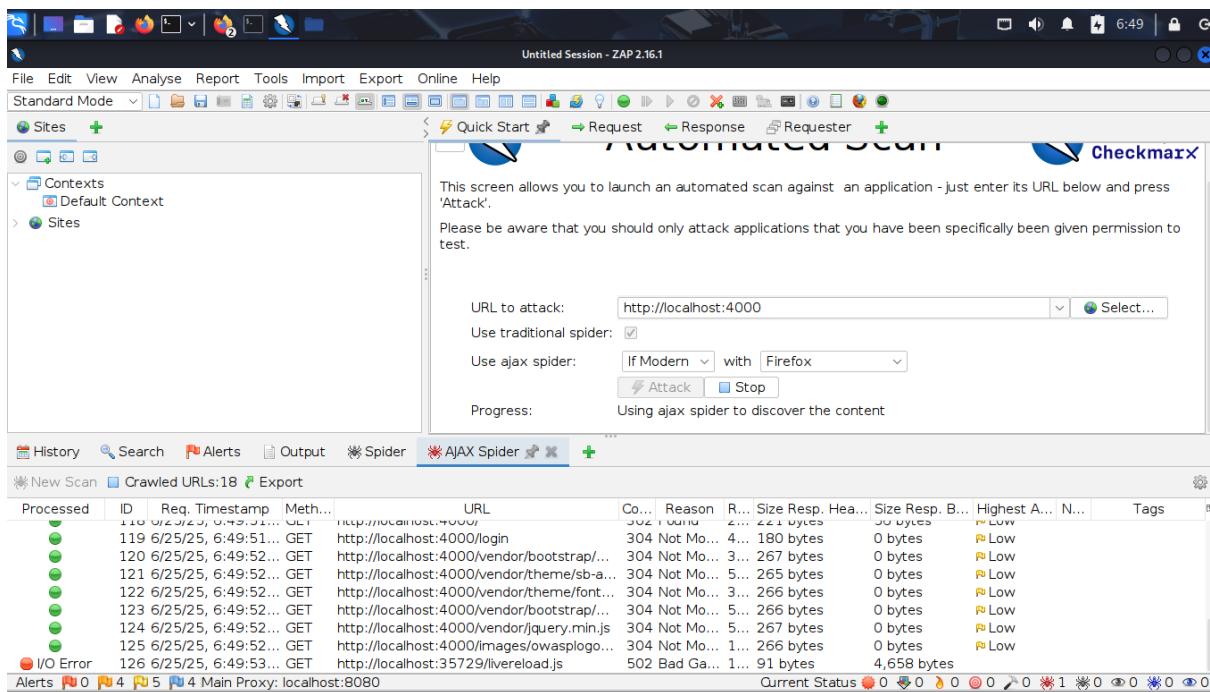
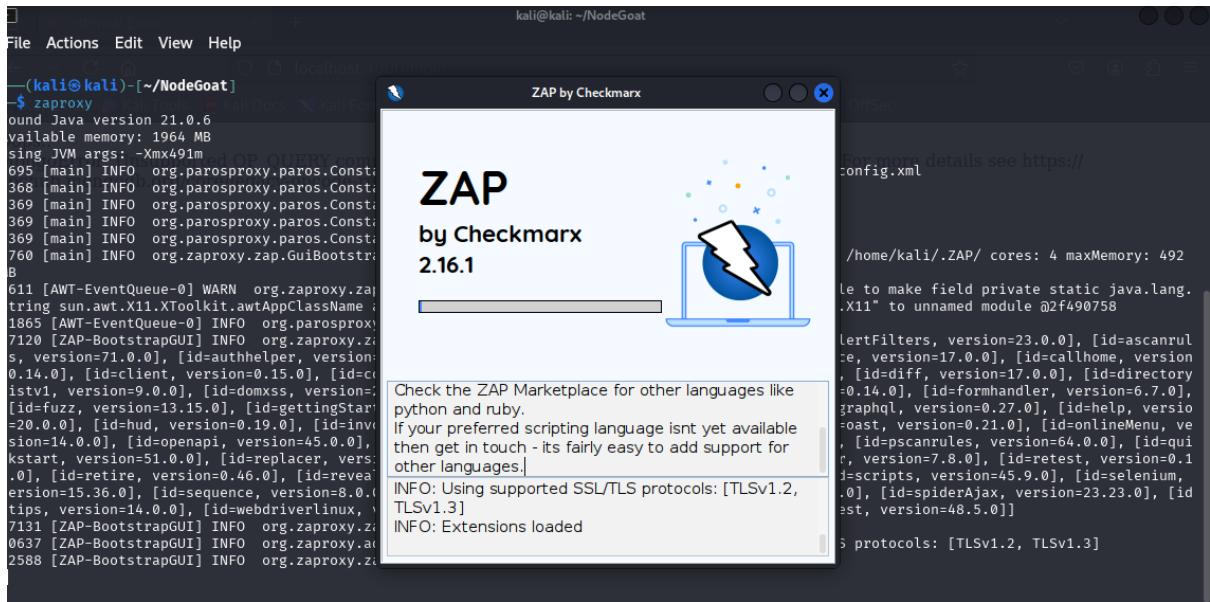
Basic Vulnerability Assessment

Tools Used: Tool	Use Case
OWASP ZAP	Auto scanning vulnerabilities
Browser DevTools	Manual XSS payloads
Manual Inputs	SQL Injection Testing



The screenshot shows a web browser window titled "OWASP Node Goat". The address bar displays "localhost:4000/login". The page content is for a login form. At the top, it says "©RetireEasy Employee Retirement Savings Management". Below that is a "User Name" field with placeholder text "Enter User Name". Below it is a "Password" field with placeholder text "Enter Password". There are two buttons at the bottom: a blue "New user? Sign Up" link and a red "Submit" button.





Vulnerabilities Discovered

ID	Vulnerability Description / Test	Type	Status
1	XSS (Cross-Site Scripting)	Injected <script>alert('XSS')</script> in Profile fields	Confirmed
2	SQL Injection	Login bypass using ' OR '1'='1 in username/password fields	Confirmed

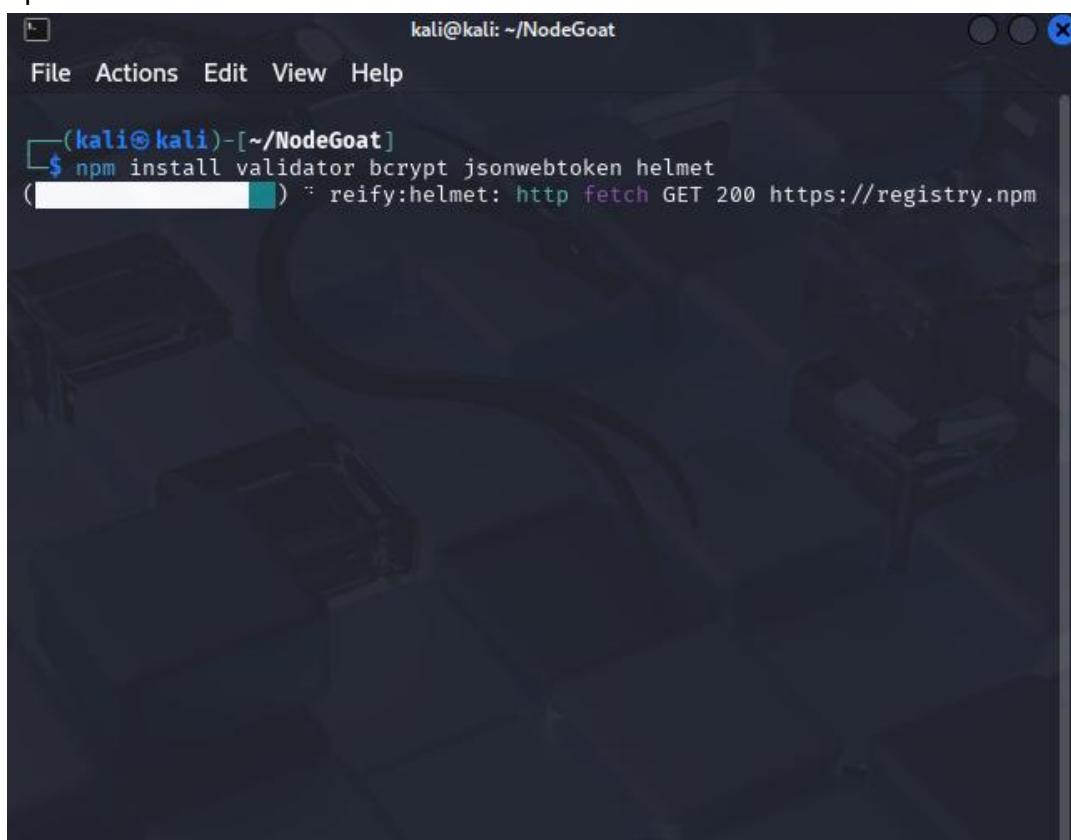
3	Weak Password Storage	Found passwords not hashed properly in database dump	Confirmed
4	Security Misconfiguration	Missing secure headers (no CSP, XFO)	Confirmed

Summary of Week 1

Tools	Status
NodeGoat Setup	<input checked="" type="checkbox"/> Complete
MongoDB Installed and Running	<input checked="" type="checkbox"/> Complete
Web App Access on Localhost	<input checked="" type="checkbox"/> Complete
Application Exploration	<input checked="" type="checkbox"/> Done
OWASP ZAP Scan	<input checked="" type="checkbox"/> Done
Manual XSS/SQLi Testing	<input checked="" type="checkbox"/> Done
Vulnerabilities Identified	<input checked="" type="checkbox"/> 4 Issues

Week 2: Fixing Vulnerabilities & Hardening the Application

- npm install validator
- npm install bcrypt
- npm install jsonwebtoken
- npm install helmet



```
kali@kali: ~/NodeGoat
File Actions Edit View Help
└─(kali㉿kali)-[~/NodeGoat]
$ npm install validator bcrypt jsonwebtoken helmet
( [██████████] )  reify:helmet: http fetch GET 200 https://registry.npm
```

Code

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

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

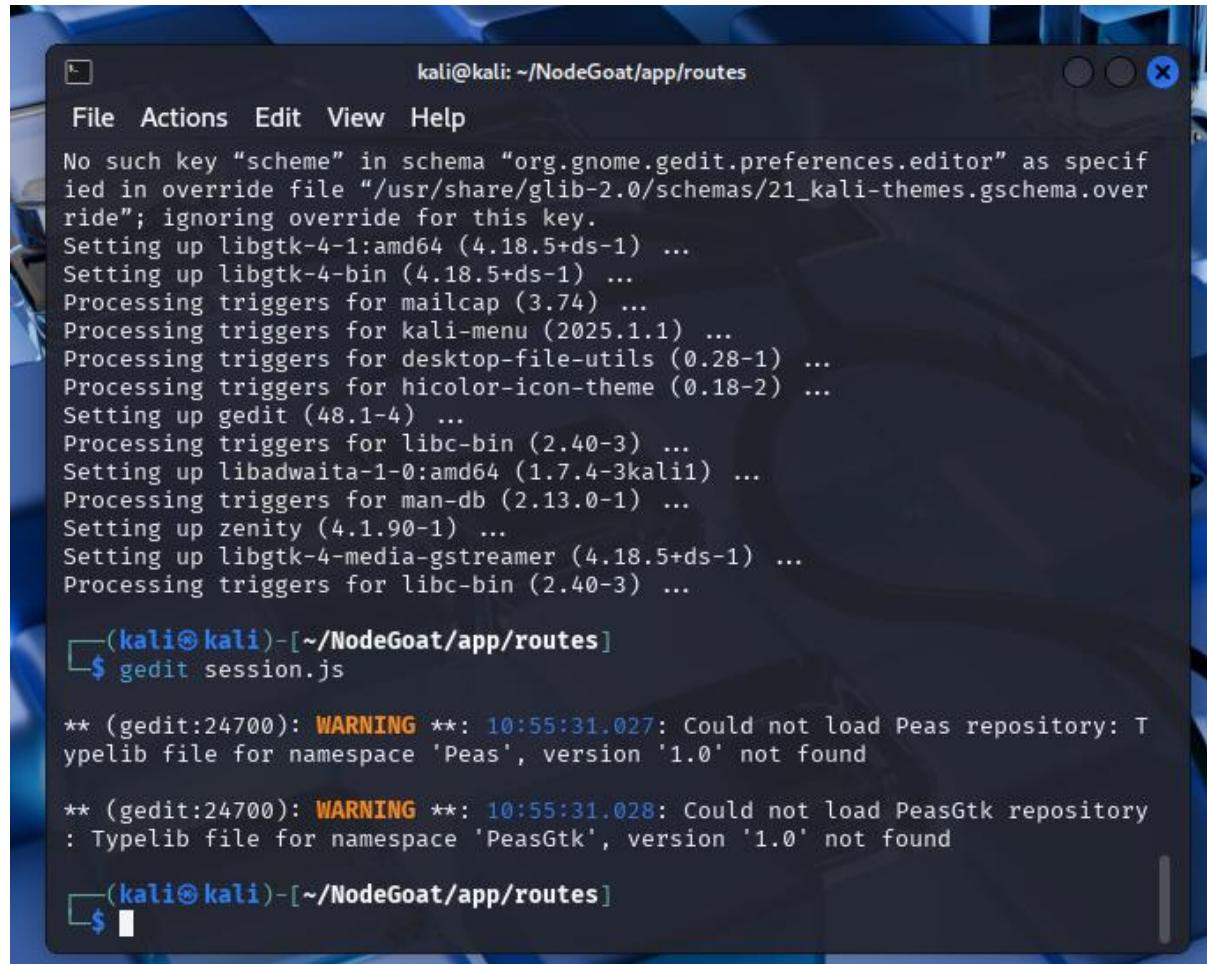
const jwt = require('jsonwebtoken');

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

res.send({ token });

const helmet = require('helmet');

app.use(helmet());
```



```
kali@kali: ~/NodeGoat/app/routes
File Actions Edit View Help
No such key "scheme" in schema "org.gnome.gedit.preferences.editor" as specified in override file "/usr/share/glib-2.0/schemas/21_kali-themes.gschema.override"; ignoring override for this key.
Setting up libgtk-4-1:amd64 (4.18.5+ds-1) ...
Setting up libgtk-4-bin (4.18.5+ds-1) ...
Processing triggers for mailcap (3.74) ...
Processing triggers for kali-menu (2025.1.1) ...
Processing triggers for desktop-file-utils (0.28-1) ...
Processing triggers for hicolor-icon-theme (0.18-2) ...
Setting up gedit (48.1-4) ...
Processing triggers for libc-bin (2.40-3) ...
Setting up libadwaita-1-0:amd64 (1.7.4-3kali1) ...
Processing triggers for man-db (2.13.0-1) ...
Setting up zenity (4.1.90-1) ...
Setting up libgtk-4-media-gstreamer (4.18.5+ds-1) ...
Processing triggers for libc-bin (2.40-3) ...

(kali㉿kali)-[~/NodeGoat/app/routes]
$ gedit session.js

** (gedit:24700): WARNING **: 10:55:31.027: Could not load Peas repository: Typelib file for namespace 'Peas', version '1.0' not found

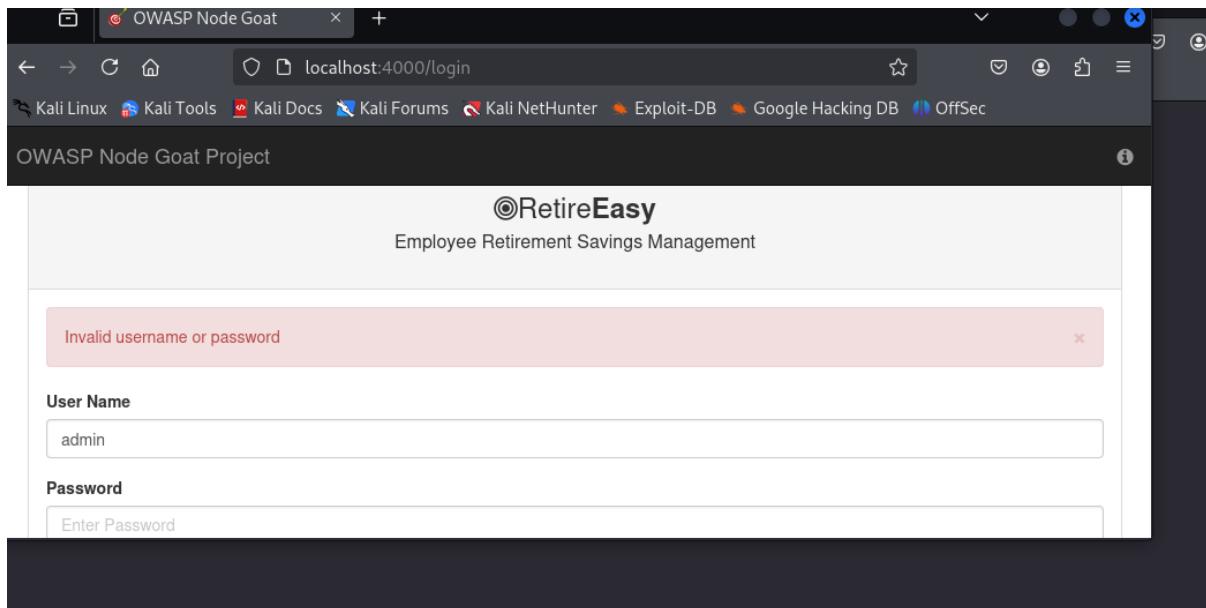
** (gedit:24700): WARNING **: 10:55:31.028: Could not load PeasGtk repository: Typelib file for namespace 'PeasGtk', version '1.0' not found

(kali㉿kali)-[~/NodeGoat/app/routes]
$
```

Screenshot of a Linux desktop environment showing two windows:

- Code Editor:** A terminal window titled "File A" showing code in session.js. The code includes imports for validator, bcrypt, and jwt, and defines a SessionHandler function that interacts with UserDAO and AllocationsDAO to handle user data.
- ZAP 2.16.1 - Untitled Session:** A web application penetration testing tool. It shows a context named "Default Context" under "Sites". The main panel displays a scan configuration for "http://localhost:4000" using an "ajax spider". The progress bar indicates "Using ajax spider to discover the content". Below the panel is a table of crawled URLs, showing entries from ID 119 to 126, all of which are marked as "Processed" and have a status of "Low" risk.

Processed	ID	Req. Timestamp	Meth...	URL	Co...	Reason	R...	Size	Re...	Hea...	Size	Re...	Highest	A...	N...	Tags
	119	6/25/25, 6:49:51...	GET	http://localhost:4000/login	304	Not Mo...	4...	180 bytes	0 bytes		0 bytes	0 bytes	Low			
	120	6/25/25, 6:49:52...	GET	http://localhost:4000/vendor/bootstrap/...	304	Not Mo...	3...	267 bytes	0 bytes		0 bytes	0 bytes	Low			
	121	6/25/25, 6:49:52...	GET	http://localhost:4000/vendor/theme/sb-a...	304	Not Mo...	5...	265 bytes	0 bytes		0 bytes	0 bytes	Low			
	122	6/25/25, 6:49:52...	GET	http://localhost:4000/vendor/theme/font...	304	Not Mo...	3...	266 bytes	0 bytes		0 bytes	0 bytes	Low			
	123	6/25/25, 6:49:52...	GET	http://localhost:4000/vendor/bootstrap/...	304	Not Mo...	5...	266 bytes	0 bytes		0 bytes	0 bytes	Low			
	124	6/25/25, 6:49:52...	GET	http://localhost:4000/vendor/jquery.min.js	304	Not Mo...	5...	267 bytes	0 bytes		0 bytes	0 bytes	Low			
	125	6/25/25, 6:49:52...	GET	http://localhost:4000/images/owasplogo...	304	Not Mo...	1...	266 bytes	0 bytes		0 bytes	0 bytes	Low			
I/O Error	126	6/25/25, 6:49:53...	GET	http://localhost:35729/livereload.js	502	Bad Ga...	1...	91 bytes	4,658 bytes							



Week 2 Summary: Vulnerability Fixing and Security Enhancements

During Week 2, I focused on fixing the vulnerabilities identified during Week 1 and implementing key security improvements to harden the OWASP NodeGoat web application.

Key Tasks Completed:

1. Input Validation and Sanitization:

I used the validator.js library to validate and sanitize all user inputs to prevent XSS and injection attacks. This ensured that only properly formatted data entered the application.

2. Password Hashing:

I integrated bcrypt for securely hashing user passwords before storing them in the database. This prevents password disclosure even if the database is compromised.

3. Token-based Authentication:

I implemented JSON Web Token (JWT) authentication to securely manage user sessions. After successful login, the server generates a JWT token, which the client sends with every authenticated request.

4. Secure HTTP Headers:

I applied Helmet.js middleware to set essential HTTP headers like Content-Security-Policy (CSP), X-Frame-Options, and others to protect against common web attacks such as clickjacking and XSS.

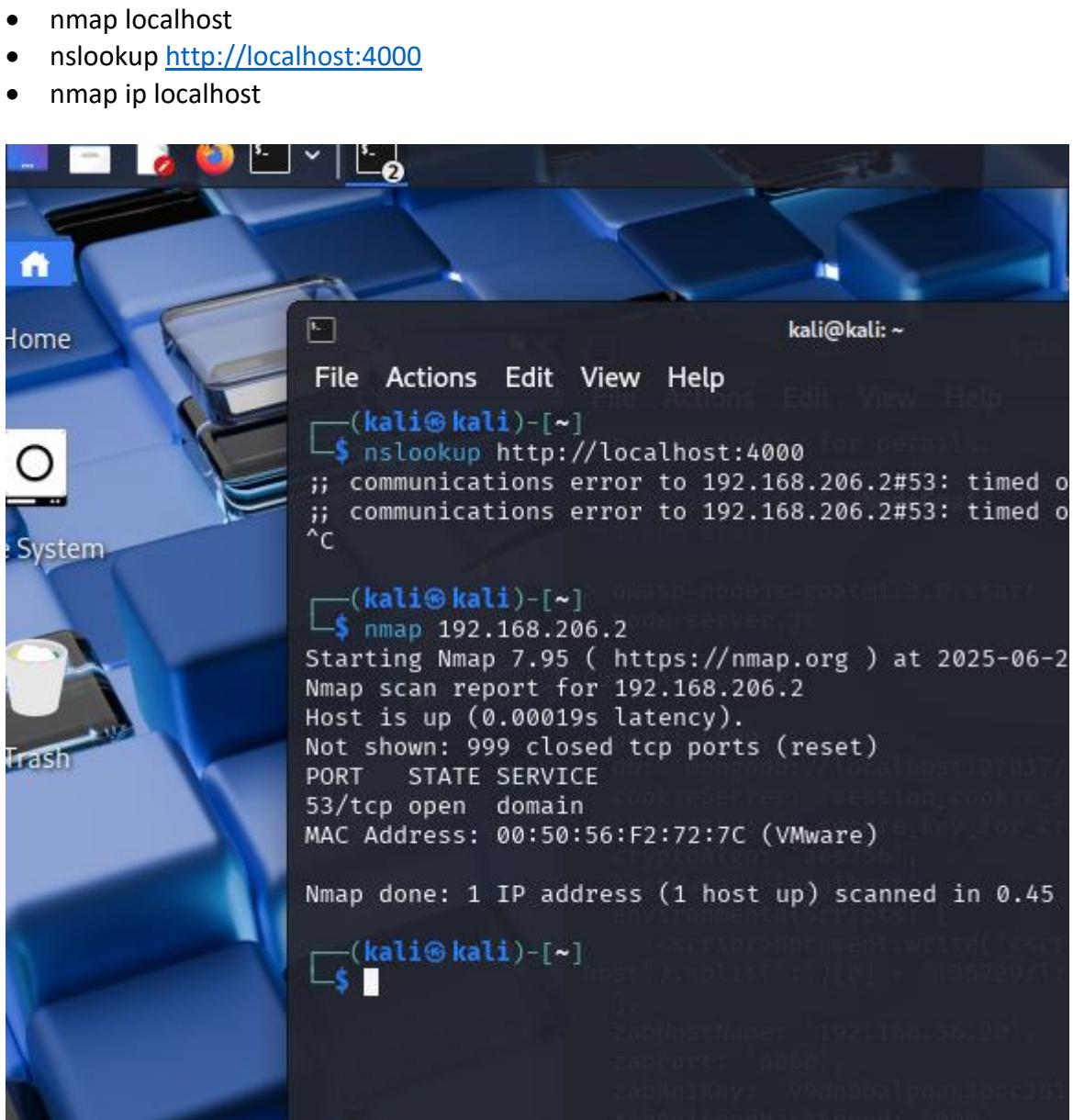
Outcome of Week 2:

- Input validation is now enforced across all user input fields.
- Passwords are no longer stored in plaintext.
- Authentication has been strengthened using JWT tokens.
- Basic HTTP security headers are now active on all server responses.

Week 3: Advanced Security Measures & Reporting

1. Basic Penetration Testing:

Performed internal network and open port scans using Nmap from Kali Linux:



The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal window title is '(kali㉿kali)-[~]'. The terminal content shows the following commands and their outputs:

```
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ nslookup http://localhost:4000
;; communications error to 192.168.206.2#53: timed out
;; communications error to 192.168.206.2#53: timed out
^C

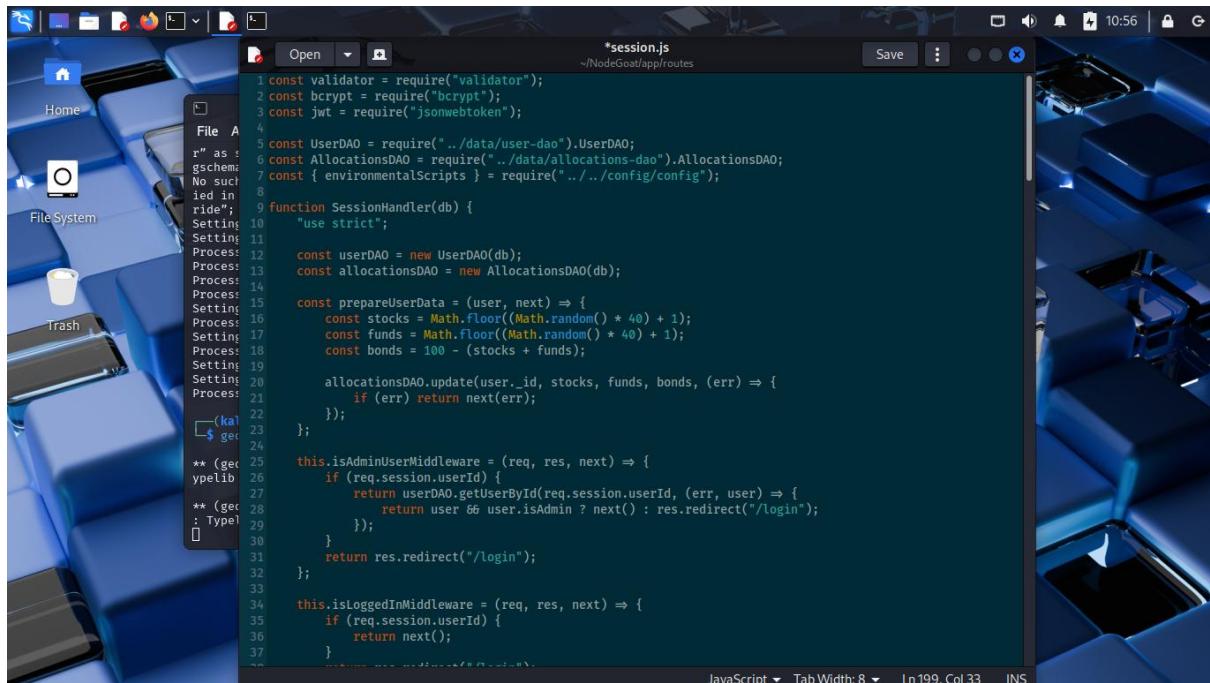
(kali㉿kali)-[~]
$ nmap 192.168.206.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-2
Nmap scan report for 192.168.206.2
Host is up (0.00019s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
53/tcp    open  domain
MAC Address: 00:50:56:F2:72:7C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.45
```

2. Setting Up Application Logging:

Installed Winston logger for application-level event tracking:

- npm install Winston



```
*session.js
const validator = require("validator");
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
const UserDAO = require("../data/user-dao").UserDAO;
const AllocationsDAO = require("../data/allocations-dao").AllocationsDAO;
const { environmentalScripts } = require("../config/config");

function SessionHandler(db) {
    "use strict";
    const userDAO = new UserDAO(db);
    const allocationsDAO = new AllocationsDAO(db);

    const prepareUserData = (user, next) => {
        const stocks = Math.floor((Math.random() * 40) + 1);
        const funds = Math.floor((Math.random() * 40) + 1);
        const bonds = 100 - (stocks + funds);

        allocationsDAO.update(user._id, stocks, funds, bonds, (err) => {
            if (err) return next(err);
        });
    };

    this.isAdminUserMiddleware = (req, res, next) => {
        if (req.session.userId) {
            userDAO.getUserById(req.session.userId, (err, user) => {
                if (err) return next(err);
                if (user.isAdmin) return next();
                return res.redirect("/login");
            });
        }
        return res.redirect("/login");
    };

    this.isLoggedInMiddleware = (req, res, next) => {
        if (req.session.userId) {
            return next();
        }
    };
}

module.exports = SessionHandler;
```

Best Practice

- ✓ Validate all user inputs
- ✓ Sanitize form fields
- ✓ Hash and salt passwords
- ✓ Use token-based authentication
- ✓ Secure HTTP headers
- ✓ Log security events
- ✓ Use HTTPS in production
- ✓ Conduct vulnerability scans regularly

Conclusion:

This cybersecurity internship project helped me gain practical experience in identifying, analyzing, and fixing real-world web application vulnerabilities. Using Kali Linux and open-source tools like OWASP ZAP, Nmap, and Node.js security libraries, I strengthened the NodeGoat application against common attacks like XSS and SQLi. Further, by implementing authentication, secure headers, and logging, I've ensured the application meets basic web security standards.