Web Application Security Assessment

Internship Task: Security Assessment of Web Application

Intern Name: Rabia Khawaja

Date: 12th April 2025

-----------------------------------------------------------------------------------------

Week 4: **Advanced Threat Detection & Web Security Enhancements**

**Goal**: Implement advanced security measures, detect threats in real-time, and secure API endpoints.

Tasks:

1. **Intrusion Detection & Monitoring: Set up real-time monitoring using Fail2Ban or OSSEC. Implement alerts for multiple failed login attempts.**

**What we need to do**

We need to detect if someone is trying to brute-force into our system and set up real-time alerts.

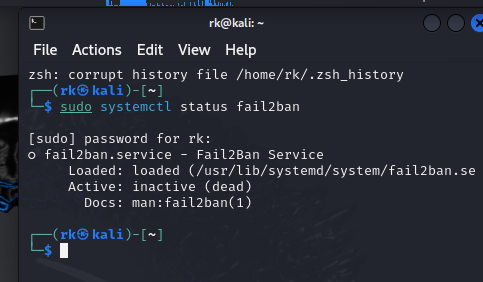
As im using Kali Linux as environment

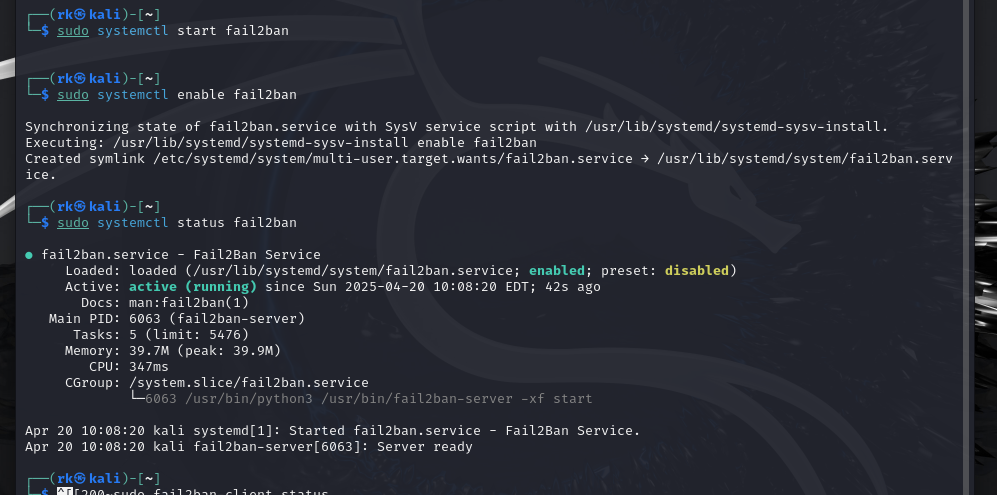
**Install Fail2Ban** for Linux systems

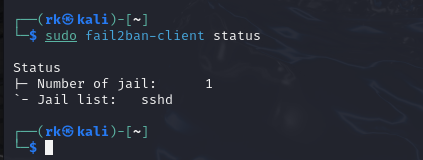
Fail2Ban monitors log files and bans IPs with suspicious behavior

· Configure /etc/fail2ban/jail.local to monitor logs like SSH or your app logs.

· Set it to ban IPs with too many failed logins.







1. **API Security Hardening: Use rate limiting with express-rate-limit to prevent brute-force attacks. Implement CORS properly to restrict unauthorized access. Use API keys or OAuth for authentication in APIs.**

**What we need to do**

Your API must:

Limit how often a user can call endpoints (rate limiting),

Block cross-origin (unauthorized) access (CORS),

Require authentication (API keys/OAuth).

**Rate Limiting** with express-rate-limit:

In terminal write

***npm install express-rate-limit***

Write **code server.js**

There write the following code

***const rateLimit = require("express-rate-limit");***

***const limiter = rateLimit({***

***windowMs: 15 \* 60 \* 1000, // 15 minutes***

***max: 100 // limit each IP to 100 requests per windowMs***

***});***

***app.use(limiter);***

Now to setup CORS

Install cors in terminal by

***npm install cors***

Open editor and write this code

***const cors = require("cors");***

***app.use(cors({***

***origin: "https://yourfrontend.com",***

***methods: ["GET", "POST"],***

***credentials: true***

***}));***

For a quick setup we can use a simple API key check.

***app.use((req, res, next) => {***

***const key = req.headers['x-api-key'];***

***if (key !== process.env.MY\_API\_KEY) return res.status(401).send('Unauthorized');***

***next();***

***});***

The complete code in vs Code, server.js will be

***const express = require('express');***

***const cors = require('cors');***

***const rateLimit = require('express-rate-limit');***

***const winston = require('winston');***

***require('dotenv').config();***

***const app = express();***

***const PORT = process.env.PORT || 3000;***

***// Configure Winston logger***

***const logger = winston.createLogger({***

***level: 'info',***

***format: winston.format.combine(***

***winston.format.timestamp(),***

***winston.format.printf(({ timestamp, level, message }) => {***

***return `${timestamp} [${level.toUpperCase()}]: ${message}`;***

***})***

***),***

***transports: [***

***new winston.transports.Console(),***

***new winston.transports.File({ filename: 'security.log' })***

***],***

***});***

***// Log startup***

***logger.info('✅ Application started');***

***// Middlewares***

***app.use(express.json());***

***// CORS Setup***

***const corsOptions = {***

***origin: ['http://localhost:3000'], // Add your frontend URL here***

***methods: ['GET', 'POST', 'PUT', 'DELETE'],***

***allowedHeaders: ['Content-Type', 'Authorization', 'x-api-key'],***

***};***

***app.use(cors(corsOptions));***

***// Rate Limiter***

***const limiter = rateLimit({***

***windowMs: 15 \* 60 \* 1000, // 15 minutes***

***max: 100, // Max 100 requests per 15 mins***

***message: '⚠️ Too many requests from this IP. Please try again later.'***

***});***

***app.use(limiter);***

***// API Key Middleware***

***const API\_KEY = process.env.API\_KEY;***

***const authenticateApiKey = (req, res, next) => {***

***const userApiKey = req.headers['x-api-key'];***

***if (userApiKey && userApiKey === API\_KEY) {***

***next();***

***} else {***

***logger.warn(`🚫 Unauthorized access attempt from IP: ${req.ip}`);***

***res.status(401).json({ message: 'Unauthorized: Invalid or missing API key' });***

***}***

***};***

***// Test Route (protected)***

***app.get('/secure-data', authenticateApiKey, (req, res) => {***

***logger.info(`🔐 Secure data accessed by IP: ${req.ip}`);***

***res.json({ message: '✅ Access granted to secure data!' });***

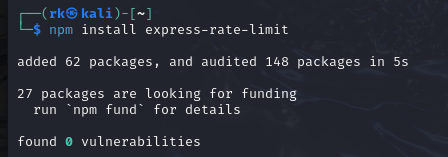
***});***

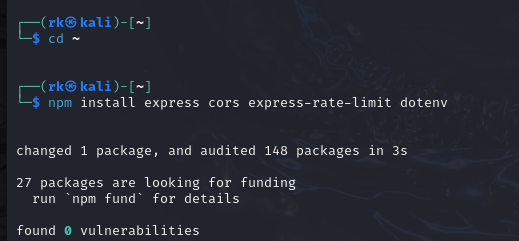
***// Server Start***

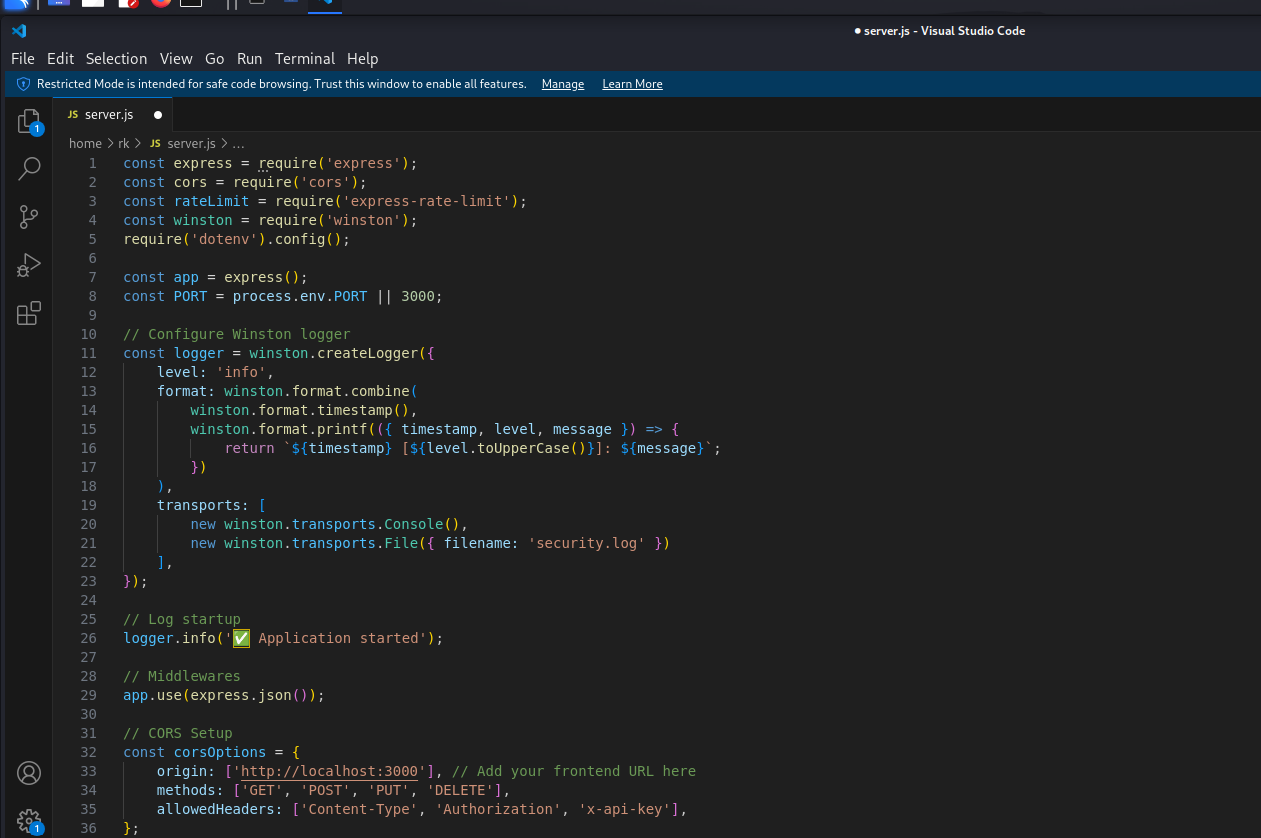
***app.listen(PORT, () => {***

***logger.info(`🚀 Server running on port ${PORT}`);***

***});***









In .env file save this,  
open the editor by nano .env

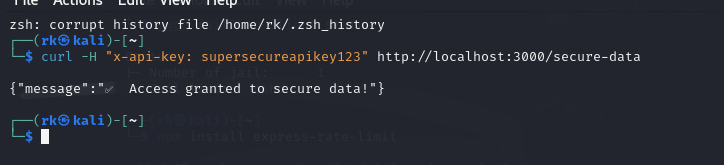
And write this

**PORT=3000**

**API\_KEY=supersecureapikey123**

Now run the server and check for route through curl

**curl -H "x-api-key: supersecureapikey123" <http://localhost:3000/secure-data>**



See the message

**“Access granted to secure data!”**

1. **Security Headers & CSP Implementation: Implement Content Security Policy (CSP) to prevent script injections. Configure Strict-Transport-Security (HSTS) for HTTPS enforcement.**

**What we need to do**

Inject security headers to block script injections and enforce HTTPS

Use helmet:

***npm install helmet***

Write this code in editor

***const helmet = require("helmet");***

***app.use(helmet());***

**Custom CSP (Content Security Policy)**:

***app.use(***

***helmet.contentSecurityPolicy({***

***directives: {***

***defaultSrc: ["'self'"],***

***scriptSrc: ["'self'", "https://trustedscripts.com"],***

***objectSrc: ["'none'"],***

***upgradeInsecureRequests: [],***

***},***

***}));***

***H*STS (Strict-Transport-Security)** is also added by Helmet, but make sure:

***app.use(***

***helmet.hsts({***

***maxAge: 31536000, // 1 year***

***includeSubDomains: true,***

***preload: true,***

***}) );***

1. Deliverables: API secured with rate-limiting and authentication. Security headers implemented with proper documentation.

GitHub repository with code updates and README.