**# 4. INPUT VALIDATION & SANITIZATION**

  // In routes/auth.js

  const { body, validationResult } = require('express-validator');

  router.post('/register',

    body('email').isEmail(),

    body('password').isLength({ min: 6 }),

    (req, res) => {

     const errors = validationResult(req);

     if (!errors.isEmpty()) return res.status(400).json({ errors: errors.array() });

     // Proceed with registration

   }

  );

**# 5. HELMET + CSP HEADERS**

// In server.js

const helmet = require('helmet');

app.use(helmet());

app.use(

  helmet.contentSecurityPolicy({

    directives: {

      defaultSrc: ["'self'"],

      scriptSrc: ["'self'", "https://apis.google.com"],

      styleSrc: ["'self'", "'unsafe-inline'"],

    },

  })

);

**# 6.RBAC MIDDLEWARE**

// middleware/role.js

module.exports = function (roles) {

  return (req, res, next) => {

    if (!roles.includes(req.user.role)) return res.status(403).json({ error: 'Access Denied' });

    next();

  };

};

**# 7. SESSION & COOKIE SETTINGS**

const cookieParser = require('cookie-parser');

app.use(cookieParser());

res.cookie('token', token, {

  httpOnly: true,

  secure: true,

  sameSite: 'strict',

  maxAge: 3600000,

});

**# 8. LOGGING WITH WINSTON + MORGAN**

const morgan = require('morgan');

const winston = require('winston');

const logger = winston.createLogger({

  transports: [

    new winston.transports.File({ filename: 'combined.log' }),

  ],

});

app.use(morgan('combined', {

  stream: {

    write: (message) => logger.info(message.trim()),

  },

}));

**# 9. DATABASE SECURITY**

// Sequelize ORM already uses parameterized queries internally

// Secure: Sequelize uses parameterized queries, preventing SQL injection

await User.findOne({ where: { email: req.body.email } });

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

**# Test Authentication Using Postman**