Week 3: Advanced Security and Final Reporting

1. Basic Penetration Testing

In the final phase of the internship, I conducted basic penetration testing to simulate common security threats and verify the effectiveness of the protective measures applied in Week 2.

- Attempted simulated attacks included:
    - Unauthorized login attempts
    - Broken authentication logic
    - Parameter tampering and bypassing input validation

These tests confirmed that the implemented security layers—including JWT authentication, input validation, and password hashing—were effectively blocking unauthorized actions, thereby enhancing the application's resilience to common attack vectors.

---

2. Set Up Basic Logging with Winston

To improve visibility into authentication activities, I configured Winston for basic logging. This allowed the system to capture and store relevant security events for auditing and monitoring.

Logging Setup:

javascript

CopyEdit

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


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

});

Example Usage During Login:

javascript

CopyEdit

```
pool.query('SELECT * FROM admin WHERE username = ?', [username], async function(err, rows) {
  if (err || rows.length === 0) {
    return res.status(400).send("User not found");
  }

  logger.info(`Login attempt by ${username} at ${new Date().toISOString()}`);
  ...
});
```

This approach allows for:

- Monitoring login attempts
- Tracking error events
- Detecting abnormal behavior (e.g., brute-force attempts)

---

3. Create a Simple Security Checklist

To ensure secure deployment, I added key best practices and verified that they are implemented correctly.

✅ Checklist Implementation Highlights:

- HTTPS Setup using self-signed certificates for encrypted communication:

javascript

CopyEdit

```
const https = require('https');
const fs = require('fs');

const httpsOptions = {
  key: fs.readFileSync('key.pem'),
  cert: fs.readFileSync('cert.pem')
};

const httpsServer = https.createServer(httpsOptions, app);
httpsServer.listen(8443, () => {
  console.log("HTTPS Server Listening on port 8443");
});
```

- **Security Best Practices Applied:**
  - User inputs are validated and sanitized
  - Passwords are hashed and salted using bcrypt
  - JWT-based authentication is enforced on all sensitive routes
  - Helmet headers applied to prevent common web vulnerabilities
  - HTTPS is used for secure communication
  - Logging is enabled for monitoring and forensic analysis
  - Production server does not leak sensitive info like version details

---

✅ Final Summary

Week 3 focused on security testing, monitoring, and enforcement of best practices:

- Penetration tests validated the application’s resistance to attacks
- Logging mechanisms were added to track and monitor login activities
- HTTPS was configured, and a security checklist was compiled for ongoing maintenance

With all three weeks completed, the project successfully fulfilled its goals of:

1. **Identifying and fixing vulnerabilities**
2. **Applying layered security measures**
3. **Validating security through testing and logging**

The application is now more secure, maintainable, and aligned with industry-standard cybersecurity practice