NOD101 – DEFENDING NODE.JS

Course Learning Objectives

Understand the security risks when developing and deploying applications in Node.js. Implement defensive coding techniques and configurations to support secure coding for Node.js.

Description

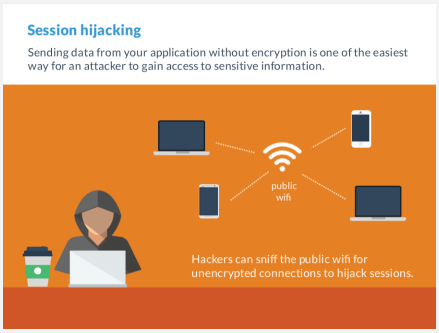
Understand the critical vulnerabilities in the OWASP Top 10 and see how these vulnerabilities can affect Node.js projects. Students will also learn to define and identify secure code, differentiate between secure coding methods, employ secure code in practice, and judge the effectiveness of these techniques.

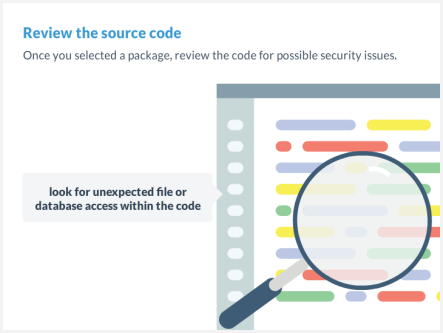
Audience

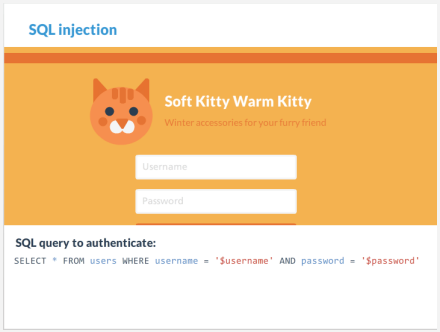
Node.js developers
Web developers

Time Required

Tailored learning - 60 minutes total







Security Compass

NOD101 – DEFENDING NODE.JS

Course Outline

1. Injection

- About the vulnerability
- SQL injection
- NoSQL injection
- Command injection
- Newsflash
- Escape query values and identifiers
- Use prepared statements
- Validate and sanitize input
- Best practices

2. Broken authentication and session management

- About the vulnerability
- Poor logout or timeout mechanisms
- Session hijacking
- Brute force password attack
- Secure the connection
- Code: Generate private key and certificate
- Code: Implement TLS to create HTTPS server
- Code: Enforce HTTP Strict Transport Security (HSTS) policy
- Set Secure cookie flag
- Session timeouts
- Brute force attack protection
- Password requirement
- Code: Generic login failure messages
- Code: Rate limiting

3. Cross-Site Scripting

- About the vulnerability
- Insecure code
- Reflected XSS attack
- Stored XSS attack
- Code: Sanitize input
- Implement Content Security Policy
- Set HttpOnly cookie flag

4. Security misconfiguration

- About the vulnerability
- Request size limits
- Implementation details leakage
- Do not run Node as root
- Set request size limits
- Hide server identity
- Disable Powered By in response header
- Hide server info in Apache
- Hide server info in NGINX
- Harden HTTP response headers on server
- Using Helmet to secure HTTP headers

5. Cross-Site Request Forgery

- About the vulnerability
- Insecure website
- The CSRF attack
- You got CSRF'ed
- POST is not safe
- How anti-CSRF tokens can protect you
- Code: Using the csurf middleware
- How our form's code looks like

6. Using components with known vulnerabilities

- About the vulnerability
- Newsflash
- Guidelines for choosing a package
- Review the source code
- Monitor dependencies for updates
- Node Security Platform
- Snyk

Security Compass