# 🛡️ WEB SECURITY ROADMAP for Full Stack Engineers

---

## 🔰 Stage 1: Foundation of Web Security

**🎯 Goal: Understand the building blocks of secure communication**

1. **What is Web Security?**
    - Why web apps are vulnerable
    - Real-life examples of breaches (e.g., Twitter, Facebook)
2. **Understanding HTTP & HTTPS**
    - Request/Response model
    - HTTPS vs HTTP
    - TLS handshake (basic overview)
3. **Encryption & Hashing**
    - Symmetric vs Asymmetric Encryption (AES, RSA)
    - One-way hashing (MD5, SHA, bcrypt, Argon2)
    - Base64 encoding (and why it's NOT security)

**✅ Your Target:**

- Be able to explain difference between encryption & hashing
- Understand why HTTPS matters and how it works internally
- Know when to use bcrypt, AES, etc.

---

## 🔐 Stage 2: Authentication (AuthN)

**🎯 Goal: Master secure login systems**

1. **Password Storage**
    - Why passwords must be hashed
    - Hashing with bcrypt and argon2
    - Salting + Peppering strategies
2. **Session-Based Auth**
    - What are sessions and how they work
    - Storing sessions in DB vs memory
    - Cookie configurations (HttpOnly, Secure, SameSite)
3. **JWT-Based Auth**
    - JWT structure (header, payload, signature)
    - How JWTs are signed & verified

    - When to use Access & Refresh Tokens
4. **Social Logins**
    - OAuth 2.0 Flow (Authorization Code Grant)
    - Google/GitHub login integrations

✅ **Your Target:**

- Create login/signup using JWT & Sessions
- Securely store passwords
- Know when to use OAuth vs email/password
- Implement token-based refresh logic

---

## 🧾 Stage 3: Authorization (AuthZ)

🎯 **Goal: Ensure users only access what they're allowed to**

1. **RBAC (Role-Based Access Control)**
    - Define roles like Admin, User, Vendor
    - Protect routes with role-checking middleware
2. **Ownership Validation**
    - Only allow owners to update/delete their data
    - Check req.user.id === resource.ownerId
3. **Attribute-Based Access Control (ABAC) (Bonus)**
    - Rule-based access (age, location, subscription level)

✅ **Your Target:**

- Implement RBAC & ownership-based access
- Secure admin-only routes and actions
- Understand the difference between RBAC and ABAC

---

## 🔥 Stage 4: Attacks & Prevention (OWASP Top 10)

🎯 **Goal: Learn vulnerabilities and how to fix them**

1. **XSS (Cross-Site Scripting)**
    - Reflected vs Stored XSS
    - Prevent with sanitization and escaping (DOMPurify)
2. **SQL Injection**
    - Traditional & modern injection
    - Prevention using ORM/Prisma, parameterized queries
3. **CSRF (Cross-Site Request Forgery)**
    - How CSRF works
    - CSRF tokens vs SameSite cookies

4. **Other OWASP Top 10**
    - Insecure Deserialization
    - Broken Access Control
    - Misconfiguration (headers, error messages)
    - Sensitive Data Exposure
    - Rate Limiting & DoS protection

✅ **Your Target:**

- Be able to explain & demo XSS, CSRF, SQLi
- Know how to defend against each with practical tools
- Understand the purpose of OWASP Top 10

---

## 🧠 Stage 5: Input Validation & Data Sanitization

🎯 **Goal: Make your app safe from untrusted data**

1. **Frontend vs Backend Validation**
    - Why backend validation is a must
2. **Schema Validators**
    - Zod, Joi, Yup – How to use them
    - Define and reuse validation schemas
3. **Avoiding Injection Attacks**
    - Escape special characters
    - Never trust user inputs

✅ **Your Target:**

- Write strong backend validation rules using Zod or Joi
- Sanitize user input for HTML/JS/DB
- Handle edge cases like file uploads, multiline fields

---

## 🌐 Stage 6: Secure Communication & CORS

🎯 **Goal: Protect APIs across origins**

1. **CORS (Cross-Origin Resource Sharing)**
    - What is CORS and why it exists
    - Preflight Requests
    - Safe & unsafe headers/methods
2. **Securing Your CORS Config**
    - When to allow credentials
    - How to restrict origins properly
3. **Content Security Policy (CSP)**

    ○ What is CSP?
    ○ Block inline scripts, restrict external content

✅ **Your Target:**

- Configure proper CORS setup in Express
- Write a basic CSP header
- Explain preflight request flow

---

## 🧱 Stage 7: Secure Deployment (Infra & DevOps Basics)

🎯 **Goal: Prevent leaks, secrets, and open ports**

1. **Environment Variable Management**
    ○ .env best practices
    ○ Avoid committing secrets in Git
2. **CI/CD Secrets**
    ○ Secrets in GitHub Actions or Vercel
    ○ Avoid exposing tokens via logs
3. **Docker Security (Optional)**
    ○ Use non-root user in Dockerfiles
    ○ Limit exposed ports

✅ **Your Target:**

- Keep secrets secure
- Prevent accidental info leaks in deployment
- Use HTTPS in prod

---

## 🛠️ Stage 8: Secure APIs

🎯 **Goal: Secure your REST APIs**

1. **Rate Limiting & Throttling**
    ○ Avoid brute force attacks
    ○ Tools: express-rate-limit, Redis
2. **API Key Security**
    ○ When and how to use API keys
    ○ Rotate and store securely
3. **Error Handling**
    ○ Don’t expose internal stack traces
    ○ Return meaningful but generic error messages

4. **Versioning & Deprecated Routes**

✅ **Your Target:**

- Secure your APIs from brute force & abuse
- Structure safe API responses
- Add request validation and error sanitization

---

## 🧪 Stage 9: Logging, Monitoring & Alerting

🎯 **Goal: Detect and respond to security events**

1. **Logging Best Practices**
   - Log login attempts, permission errors, IPs
2. **Audit Trails**
   - Track sensitive actions like password change, role update
3. **Alerting**
   - Notify on suspicious activity (failed logins, DDoS)

✅ **Your Target:**

- Create basic logs for security events
- Maintain action history for users
- Know how to monitor app behavior

---

## 🧰 Stage 10: Tooling & Testing

🎯 **Goal: Use tools to automate security**

1. **Security Linters & Scanners**
   - ESLint security plugins
   - npm audit, Snyk
2. **Pentesting Basics**
   - Postman for testing auth flow
   - Burp Suite/ZAP for manual testing
3. **Secure Headers**
   - Helmet.js (set security headers in Express)

✅ **Your Target:**

- Scan code for vulnerabilities
- Use helmet and other middlewares
- Be able to explain how you tested your app for security