
WEEK 1 – DAY 3

WEB ARCHITECTURE (Security-Oriented, Ultra-Detailed)

---

1. Why Web Architecture Is Important for Security

Most cybersecurity roles today deal with:

- Web applications
- APIs
- Cloud services

Almost all attacks happen at the web layer:

- SQL Injection
- XSS
- CSRF
- Broken authentication
- API abuse

If you understand how a web app is **designed and communicates**, you can:

- Find vulnerabilities
- Explain attacks
- Defend systems properly

---

2. Client–Server Model (Foundation)

Simple Meaning

Web applications work on a **client–server model**.

- **Client** → Requests data
- **Server** → Processes request and sends response

Who Is the Client?

- Browser (Chrome, Firefox)
- Mobile app
- API client (Postman)

Who Is the Server?

- Web server (Apache, Nginx)
- Application server (Flask, Django, Node.js)
- Database server

Step-by-Step Example (Very Important)

1. You open www.example.com
2. Browser sends an HTTP request to server
3. Server processes the request
4. Server sends an HTTP response
5. Browser displays content

Security View

- Client is **untrusted**
- Server must **validate everything**

Interview line:

"Never trust client input. Always validate on the server."

---

3. REST APIs (Representational State Transfer)

What Is an API?

An API allows **two programs to communicate**.

REST API is the **most common web API design**.

Key REST Principles (Easy Explanation)

1. **Stateless**
    - Each request is independent
    - Server does not remember previous requests
2. **Resource-based**
    - Everything is a resource (users, products, orders)
3. **Uses HTTP methods**
    - GET, POST, PUT, DELETE

REST API Example

GET /api/users

POST /api/users

GET /api/users/5

DELETE /api/users/5

Security Risks in APIs

- Broken authentication
- IDOR (Insecure Direct Object Reference)
- Missing rate limits
- Excessive data exposure

---

4. HTTP Methods (What Action to Perform)

GET

- Used to **fetch data**
- Should not modify data

Example:

GET /profile

Security risk:

- Sensitive data in URL

POST

- Used to **send data**
- Creates resources

Example:

POST /login

Security risk:

- Injection attacks

PUT / PATCH

- Used to **update data**

DELETE

- Used to **remove data**

Interview Tip

“Correct HTTP method usage is important for both functionality and security.”

---

5. HTTP Status Codes (Server Response Meaning)

2xx – Success

- 200 OK
- 201 Created

3xx – Redirection

- 301 Moved Permanently
- 302 Found

4xx – Client Errors

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found

5xx – Server Errors

- 500 Internal Server Error
- 502 Bad Gateway

Security Insight

Too detailed error messages can help attackers.

---

6. Cookies vs Headers (Very Important for Security)

6.1 Cookies

What Are Cookies?

Small data stored in the browser.

Used for:

- Sessions
- Login state
- User tracking

Example Cookie

session_id=abc123

Security Risks

- Session hijacking
- XSS stealing cookies

Secure Cookie Flags

- HttpOnly
- Secure
- SameSite

6.2 Headers

What Are Headers?

Metadata sent with requests and responses.

Examples:

- Authorization
- Content-Type
- User-Agent

Example Authorization Header

Authorization: Bearer <token>

Security Advantage

- Tokens in headers are safer than cookies (in many cases)

---

7. Cookies vs Headers (Comparison)

Feature	Cookies	Headers
Stored in browser	Yes	No
Auto sent	Yes	No
Used for sessions	Yes	Yes
CSRF risk	High	Low

---

8. Hands-On: Inspect Requests in Browser DevTools

Why This Matters

Understanding requests visually helps you:

- Debug issues
- Spot vulnerabilities
- Learn how attacks work

Step-by-Step (Chrome)

1. Open a website
2. Press **F12**
3. Go to **Network** tab
4. Refresh page
5. Click any request

What to Observe

- Request URL

- HTTP method
- Headers
- Cookies
- Response status code

Interview Tip

“I regularly inspect requests using browser DevTools to understand application behavior.”