AppSec Course Priority Sequence

Aligned to a Certification Progression (PortSwigger → OSWA → BSCP/OSWE)

Prepared for: **Jordan Suber**

January 19, 2026

Purpose. This document provides a practical, priority-ordered sequence for the provided PortSwigger Web Security Academy learning paths, structured to align with a progressive certification-style roadmap:

- **Phase 1 (Foundation):** high-yield core web security concepts and exploitation patterns.
- **Phase 2 (OSWA alignment):** breadth + workflow that strengthens assessment performance.
- **Phase 4 (Depth):** modern surfaces and higher-complexity vulnerability classes, consistent with advanced practitioner expectations.

Scope. The sequence uses only the learning paths explicitly provided in the source list. A short section at the end identifies *important gaps* that are commonly required for comprehensive alignment but are not present in the excerpted list.

1 Input Learning Paths Included

The following learning paths (as provided) are included in this prioritization:

- Web cache deception (Practitioner)
- WebSockets vulnerabilities (Practitioner)
- Authentication vulnerabilities (Practitioner)
- Server-side request forgery (SSRF) attacks (Practitioner)
- Prototype pollution (Practitioner)
- Clickjacking (UI redressing) (Practitioner)
- GraphQL API vulnerabilities (Practitioner)
- Cross-origin resource sharing (CORS) (Practitioner)
- Path traversal (Practitioner)
- NoSQL injection (Practitioner)
- Race conditions (Practitioner)

- Cross-site request forgery (CSRF) (Practitioner)
- File upload vulnerabilities (Practitioner)
- Web LLM attacks (Practitioner)
- API testing (Practitioner)
- Server-side vulnerabilities (Apprentice)
- SQL injection (Practitioner)

2 Priority Sequence by Phase

2.1 Phase 1 (Foundation): High-yield core AppSec fundamentals

Goal: Build the core vulnerability "muscle memory" and web security mental models needed before certification-grade assessments.

1. **Server-side vulnerabilities (Apprentice)**
   Broad orientation to common server-side issues and real-world attacker workflows.
2. **SQL injection**
   Canonical injection class; builds request/response reasoning, data exfiltration patterns, and secure query defense principles.
3. **Authentication vulnerabilities**
   High-impact and foundational to real application security; improves ability to reason about sessions, credential workflows, and identity controls.
4. **Path traversal**
   Common, high severity, and excellent for learning systematic input manipulation and server behavior inference.
5. **File upload vulnerabilities**
   Practical exploitation and defense patterns; complements traversal and authentication by exercising validation, storage, and execution boundaries.
6. **Server-side request forgery (SSRF) attacks**
   Modern, high-impact issue in cloud and microservice environments; develops internal network and metadata endpoint attack intuition.
7. **NoSQL injection**
   Extends injection reasoning to modern stacks (operators, schema-less edge cases, query manipulation).
8. **Cross-site request forgery (CSRF)**
   Core web risk; strengthens understanding of browser security model, state-changing requests, and anti-CSRF defenses.
9. **Cross-origin resource sharing (CORS)**
   Critical modern SPA/API boundary topic; directly supports robust security review of cross-origin configurations.

10. **Clickjacking (UI redressing)**
    Efficient coverage item; reinforces UI-layer protections and defense-in-depth.

Phase 1 milestone. After completing items 1–10, you should be equipped with the baseline breadth expected for moving into OSWA-aligned practice and higher-intensity assessment preparation.

2.2 Phase 2 (OSWA alignment): Assessment breadth and testing workflow

Goal: Expand beyond core vuln classes into attack surface discovery and practical methodology.

11. **API testing**
    Directly supports endpoint discovery, recon, and parameter-handling skill—especially relevant for modern service-oriented architectures.

12. **Race conditions**
    Increasingly common; strengthens concurrent behavior analysis and tool-driven exploitation workflows (e.g., Repeater/Turbo Intruder patterns).

2.3 Phase 4 (Depth): Modern surfaces and higher-complexity vulnerability classes

Goal: Build deeper expertise consistent with advanced practitioner expectations and broader certification pathways.

13. **GraphQL API vulnerabilities**
    Modern API surface area with distinctive failure modes; improves schema/resolver reasoning and bypass identification.

14. **WebSockets vulnerabilities**
    Real-time application security; extends methodology beyond classic request/response paradigms.

15. **Prototype pollution**
    Higher complexity; valuable for JavaScript-heavy stacks and for translating findings into secure coding guidance.

16. **Web cache deception**
    Niche but high-value; sharpens understanding of caching boundaries, routing, and origin-versus-cache discrepancies.

2.4 Emerging / Specialized (Optional, after strong fundamentals)

17. **Web LLM attacks**
    Highly relevant for organizations deploying LLM-enabled features; best taken after core web and API foundations are strong, unless your current product surface makes this immediately applicable.

3 Summary Table

Phase	**Learning Paths (Priority Order)**
Phase 1 (Foundation)	Server-side vulnerabilities; SQL injection; Authentication vulnerabilities; Path traversal; File upload vulnerabilities; SSRF attacks; NoSQL injection; CSRF; CORS; Clickjacking
Phase 2 (OSWA alignment)	API testing; Race conditions
Phase 4 (Depth)	GraphQL API vulnerabilities; WebSockets vulnerabilities; Prototype pollution; Web cache deception
Emerging / Specialized	Web LLM attacks

4 Important Gaps (Not Included in the Provided Excerpt)

If the objective is strict alignment to a comprehensive web application security preparation track, ensure you also cover high-frequency topics that are commonly required but were *not present* in the supplied list. Examples typically include:

- Cross-site scripting (XSS)
- Access control / IDOR
- Request smuggling
- XML external entity (XXE)
- Deserialization / object injection

Recommendation. Treat the gaps above as a parallel "coverage checklist." If your certification roadmap explicitly includes them, integrate them into Phase 1–2 based on the guide's ordering and your organization's stack (e.g., SPA-heavy, GraphQL-heavy, microservices-heavy).

5 Suggested Usage

- Use **Phase 1** as your baseline competency build.
- Use **Phase 2** to raise assessment performance through methodology and coverage expansion.
- Use **Phase 4** for depth and modern surface expertise, emphasizing realistic attack paths and defense strategies.

Document version: v1.0