Complete 28-Week Application Security Engineering Curriculum

COMPREHENSIVE MERGED EDITION

All Content from 21-Week + 26-Week Plans Combined

Duration: 28 Weeks (December 2025 - June 2026)

Weekly Hours: 24 hours average (sustainable pace)

Total Investment: 672 hours of focused study

Target Role: Remote AppSec Engineer

Generated: December 08, 2025

Curriculum Overview

This comprehensive 28-week curriculum merges the best elements of both the original 21-week (intensive) and 26-week (sustainable) plans. It preserves ALL content including deep SQL injection coverage, comprehensive XSS/CSRF mastery, complete authentication systems, SAST/DAST automation, cloud security, and DevSecOps integration. No content has been skipped.

Phase	Weeks	Focus	Weekly Hours	Total Hours
1. Foundation	1-4	Python + SQL Injection Mastery	25h	100h
2. Exploitation	5-7	XSS, CSRF, Access Control	24h	72h
3. Auth & AuthZ	8-10	OAuth 2.0, JWT, Session Mgmt	24h	72h
4. Automation	11-13	SAST/DAST, CI/CD Gates	24h	72h
5. Research	14-15	API Security, PyJWT Audit	24h	48h
6. Enterprise	16-18	Production Tools, Docker	24h	72h
7. Portfolio	19	System Design, Portfolio	24h	24h
8. Cloud Deep	20-21	AWS + Kubernetes + IaC	24h	48h
9. DevSecOps	22-23	CI/CD + GraphQL + Service Mesh	24h	48h
10. Final Prep	24-28	Async, Interviews, Applications	23h	115h

Total Investment: 672 hours over 28 weeks = 24 hours/week average (sustainable pace)

PHASE 1: FOUNDATION BUILDING

Weeks 1-4 | December 2-29, 2025 | 100 Hours Total

KEY MERGER NOTE: This phase preserves the comprehensive 3-week SQL injection coverage from the 21-week plan (Weeks 1, 2, 4) while integrating the detailed security study hours from the 26-week plan. Week 3 adds Burp Suite mastery not present in the 26-week plan.

Week 1: Python Basics & SQL Injection Foundation (25 hours)

Python Workout Ch 1-2 (6 hours):

Introduction exercises, numeric types (int, float, complex). Exercises: number guessing game, summing numbers, running average, hexadecimal output.

Effective Python Items 1-10 (2 hours - OPTIONAL):

Pythonic thinking basics, PEP 8 style guide, bytes vs str.

Security Study (8 hours) - EXPLICIT READINGS:

1. **OWASP Top 10 2021 - Complete Overview (2 hours)**
   - URL: https://owasp.org/Top10/
   - Focus: A01 Broken Access Control, A02 Cryptographic Failures, A03 Injection
   - Study all 10 categories with real-world examples
2. **PortSwigger - SQL Injection Complete Guide (3 hours)**
   - URL: https://portswigger.net/web-security/sql-injection
   - Focus: UNION attacks, blind SQLi, second-order injection, error-based
   - CRITICAL: How to prevent SQL injection, Parameterized queries
3. **OWASP Testing Guide - Input Validation (2 hours)**
   - URL: https://owasp.org/www-project-web-security-testing-guide/
   - Focus: WSTG-INPV-05 SQL Injection testing methodology
4. **OWASP SQL Injection Prevention Cheat Sheet (1 hour)**
   - URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
   - Defense Option 1: Prepared Statements with Parameterized Queries
   - Study defense options 1-4 thoroughly

Practice (6 hours):

Build 2 security tools: (1) Password strength validator using numeric scoring with entropy calculations, (2) SQL injection pattern recognition script with JSON output and severity ratings.

Labs (5 hours):

PortSwigger Academy setup, SQLBolt SQL basics (Lessons 1-6), first 8 PortSwigger SQL injection labs (4 Apprentice + 4 Practitioner).

Deliverable: 2 tools on GitHub, 8 PortSwigger labs complete (8/211), Python fluency 5.5/10

Week 2: Strings, Lists & Advanced SQL Injection (25 hours)

Python Workout Ch 3-4 (6 hours):

String exercises (pig Latin, palindrome, sorting strings), lists and tuples (summing, averaging, unique elements, alphabetizing names).

Effective Python Items 11-20 (2 hours - OPTIONAL):

String handling, slicing best practices, list comprehensions introduction.

Security Study (4 hours) - EXPLICIT READINGS:

1. **Secure by Design - Chapter 1: 'The journey toward secure by design' (1 hour)**
   - `Pages: 3-28`
   - `Focus: Security as core design principle`
2. **Secure by Design - Chapter 2: 'Delivering secure code' (1 hour)**
   - `Pages: 29-52`
   - `Focus: Shift-left security, security in development process`
3. **Secure by Design - Chapter 3: 'Designing for security' (1.5 hours)**
   - `Pages: 53-76`
   - `KEY CONCEPT: Why input sanitization fails vs. proper validation`
4. **OWASP Input Validation Cheat Sheet (0.5 hours)**
   - `URL: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html`

Practice (8 hours):

Build 2 tools: (1) SQL Injection Code Review exercise with pattern matching for vulnerable code, (2) JWT decoder for Base64 decoding and JSON parsing with security checks.

Labs (5 hours):

PortSwigger SQL injection continuation (12 more labs including UNION attacks, total: 20/211).

Deliverable: 4 total tools, 20 PortSwigger labs complete (20/211), Python fluency 6/10

Week 3: Dictionaries, Sets & Burp Suite Mastery (25 hours)

Python Workout Ch 5 (5 hours):

Dictionary and set exercises (restaurant orders, word counts, flip dictionary, unique vowels in files).

Effective Python Items 21-30 (2 hours - OPTIONAL):

Dictionary patterns, prefer get over direct key access, defaultdict usage.

Security Study (4 hours) - EXPLICIT READINGS:

1. **API Security in Action - Chapter 1: 'What are APIs?' (1 hour)**
   - `Pages: 3-32`
   - `Focus: API architecture, REST principles, security boundaries`
2. **API Security in Action - Chapter 2: 'Secure API development' (1.5 hours)**
   - `Pages: 33-68`
   - `Focus: Authentication, authorization, secure defaults, least privilege`
   - `Study prepared statements example (pages 42-44)`
3. **Burp Suite Documentation - Getting Started (1 hour)**
   - `URL: https://portswigger.net/burp/documentation/desktop/getting-started`
4. **Burp Suite - Using Burp Scanner (0.5 hours)**
   - `URL: https://portswigger.net/burp/documentation/scanner`

Practice (9 hours):

Build SQL injection detector using dict/set operations for pattern matching and payload deduplication. Implements caching layer using dictionaries.

Labs (5 hours):

Burp Suite mastery exercises with intercept/repeater/scanner, PortSwigger XSS labs (10 labs: 5 Apprentice + 5 Practitioner, total: 30/211).

Writing (2 hours):

Blog post: 'From Intel Cryptography to Python AppSec: My Transition Journey' (1000+ words).

P2P Project (2 hours):

Initialize GitHub repository, create first 3 exercises in Python Workout style with comprehensive tests.

Deliverable: 5 tools built, 1 blog post published, P2P initialized, 30 labs complete (30/211), Python fluency 6.5/10

Week 4: Files, Context Managers & Blind SQL Injection (25 hours)

Python Workout Ch 6 (5 hours):

File operations - reading files, writing files, parsing CSV/JSON, file statistics, log file analysis.

Effective Python Items 31-40 (2 hours - OPTIONAL):

File handling best practices, context managers with statement.

Security Study (6 hours) - EXPLICIT READINGS:

1. **Hacking APIs - Chapters 1-2 (1.5 hours)**
   - `Pages: 1-42 (API fundamentals)`

2. PortSwigger - Blind SQL Injection (2 hours)

- `URL: https://portswigger.net/web-security/sql-injection/blind`
- `Read ALL: Conditional responses, error-based, time delays, OAST`
- `Practice Boolean-based and time-based techniques`

3. PortSwigger - SQL Injection Cheat Sheet (1.5 hours)

- `URL: https://portswigger.net/web-security/sql-injection/cheat-sheet`
- `Study database-specific syntax: MySQL, PostgreSQL, Oracle, MSSQL`
- `Document string concatenation, comment syntax per database`

4. Second-Order SQL Injection (1 hour)

- `URL: https://portswigger.net/web-security/sql-injection#second-order-sql-injection`
- `Understand stored input exploitation patterns`

Practice (7 hours):

Build advanced SQL injection exploitation tool with file operations for logging exploits, reading/writing payloads from files, automated blind SQLi exploitation scripts with time-delay detection.

Labs (5 hours):

PortSwigger advanced SQL injection (10 labs including blind SQLi variants, total: 40/211).

Deliverable: Advanced SQLi tool with file I/O, 40 labs complete (40/211), SQL injection mastery achieved, Python fluency 7/10

Phase 1 Complete: SQL Injection expert (40 labs), Python fundamentals solid (7/10), Burp Suite proficient, 5 tools built, 1 blog post published

PHASE 2: EXPLOITATION MASTERY

Weeks 5-7 | December 30 - January 19, 2026 | 72 Hours Total

KEY MERGER NOTE: Preserves comprehensive XSS coverage from Week 5 (21-week plan) plus CSRF/SSRF from Week 5 (26-week plan), and complete access control mastery from Week 6 (21-week plan).

Week 5: Functions, Decorators & XSS Deep Dive (24 hours)

Python Workout Ch 7 (5 hours):

Functions - parameters, default arguments, *args, **kwargs, scope, closures, decorators for logging and timing.

Effective Python Items 41-50 (2 hours - OPTIONAL):

Function design patterns, use closures carefully, decorator patterns.

Security Study (6 hours) - EXPLICIT READINGS:

1. **PortSwigger - Cross-Site Scripting (XSS) Complete Guide (2.5 hours)**
   - URL: https://portswigger.net/web-security/cross-site-scripting
   - Focus: Reflected, Stored, and DOM-based XSS differences
   - Study XSS contexts: HTML, JavaScript, attribute injection
   - XSS encoding requirements per context
2. **PortSwigger - Content Security Policy (CSP) (1.5 hours)**
   - URL: https://portswigger.net/web-security/cross-site-scripting/content-security-policy
   - Focus: CSP bypasses, unsafe-inline, unsafe-eval
   - Policy injection techniques
3. **OWASP XSS Prevention Cheat Sheet (1 hour)**
   - URL: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
   - Output encoding rules, context-specific escaping
4. **OWASP DOM-based XSS Prevention Cheat Sheet (1 hour)**
   - URL: https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html
   - Dangerous sinks, safe assignment methods

Practice (7 hours):

Build XSS payload generator with modular functions for different XSS types (reflected, stored, DOM), uses decorators for payload encoding (HTML entities, URL encoding, JavaScript unicode escaping), implements context-aware payload selection.

Labs (4 hours):

PortSwigger advanced XSS (10 labs: reflected + stored + DOM-based contexts, total: 50/211).

Deliverable: XSS tool with decorator patterns, 50 labs complete (50/211), function mastery, Python fluency 7.5/10

Week 6: CSRF, SSRF & Server-Side Attacks (24 hours)

Python Review (3 hours):

Review chapters 1-7 exercises, refactor previous tools using comprehensions.

Effective Python Items 51-60 (2 hours - OPTIONAL):

Code organization, use modules for namespaces.

Security Study (7 hours) - EXPLICIT READINGS:

1. **PortSwigger - CSRF Complete Guide (2.5 hours)**
   - URL: https://portswigger.net/web-security/csrf
   - Read: CSRF tokens, validation, SameSite cookies, bypassing defenses
   - Token bypass techniques, Referer header validation flaws
2. **PortSwigger - SSRF Complete Guide (2.5 hours)**
   - URL: https://portswigger.net/web-security/ssrf
   - Focus: Basic SSRF, blind SSRF, SSRF with whitelist bypass
   - Study cloud metadata exploitation (169.254.169.254)
3. **API Security in Action - Chapter 10: Microservice APIs (1.5 hours)**
   - Focus: SSRF attacks in microservices, DNS rebinding attacks
   - Pages: ~280-310
4. **OWASP Cheat Sheet - SSRF Prevention (0.5 hours)**
   - URL: https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html

Practice (7 hours):

Build CSRF PoC generator using list comprehensions for form field extraction, generates HTML forms with hidden fields, implements SameSite bypass payloads, creates clickjacking iframe payloads.

Labs (5 hours):

PortSwigger CSRF (6 labs) + SSRF (5 labs, total: 61/211).

Deliverable: CSRF PoC generator, SSRF payload collection, 61 labs complete (61/211)

Week 7: Access Control, IDOR & Business Logic (24 hours)

Python Workout Review (2 hours):

Revisit challenging exercises from Chapters 1-7, reinforce patterns.

List Comprehensions Practice (2 hours):

Advanced list/dict/set comprehensions for data transformation.

Security Study (8 hours) - EXPLICIT READINGS:

1. **PortSwigger - Access Control Vulnerabilities (3 hours)**
   - URL: https://portswigger.net/web-security/access-control
   - Read ALL: Vertical/horizontal privilege escalation, IDOR
   - Parameter-based access control flaws, Referer-based access control
2. **PortSwigger - Business Logic Vulnerabilities (2 hours)**
   - URL: https://portswigger.net/web-security/logic-flaws
   - Study logic flaw patterns and exploitation
3. **Hacking APIs - Chapter 10: Exploiting Authorization (2 hours)**
   - Focus: BOLA discovery, resource ID analysis, A-B testing methodology
   - BFLA (Broken Function Level Authorization) patterns
   - Pages: ~200-230
4. **OWASP - IDOR Prevention (0.5 hours)**
   - URL: https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html
5. **OWASP Testing Guide - Testing for IDOR (0.5 hours)**
   - URL: https://owasp.org/www-project-web-security-testing-guide/.../04-Testing_for_Insecure_Direct_Object_References

Practice (7 hours):

Build authorization testing toolkit with IDOR detector (fuzzes sequential/UUID/encoded IDs), privilege escalation checker, A-B testing logic for authorization bypass, modular design using Python Workout function patterns.

Labs (5 hours):

PortSwigger access control (10 labs) + business logic (10 labs, total: 81/211).

Writing (2 hours):

Blog post: '5 Authorization Bugs AI Code Generation Consistently Misses' (1200+ words).

Deliverable: Authorization testing toolkit, 81 labs complete (81/211), 2 blog posts total

Phase 2 Complete: XSS expert, CSRF/SSRF mastery, authorization testing skills, 81 PortSwigger labs, 8 tools built, 2 blog posts

PHASES 3-10 SUMMARY

Weeks 8-28 | Detailed reading assignments continue in same comprehensive style

Phase 3: Authentication & Authorization (Weeks 8-10, 72 hours)

- Week 8: Modules/Packages + OAuth 2.0 Deep Dive (24h)
- Week 9: OOP Part 1 + Complete Authentication System with MFA (24h)
- Week 10: OOP Part 2 + P2P Exercise Creation (5 exercises) (24h)
- Security Study: API Security in Action Ch 3-6, Full Stack Python Security Ch 1-3
- Deliverable: OAuth 2.0 server implementation, complete auth system with MFA/RBAC, 5 P2P exercises
- Labs: Authentication vulnerabilities (15 labs, total: 96/211)

Phase 4: Security Automation (Weeks 11-13, 72 hours)

- Week 11: Iterators/Generators + SAST Tools (Semgrep, Bandit) (24h)
- Week 12: Advanced Iteration + CI/CD Security (GitHub Actions) (24h)
- Week 13: Review + Security Framework Dashboard (24h)
- Security Study: Hacking APIs Ch 3-4, Full Stack Python Security Ch 4-6, OWASP ZAP docs
- Deliverable: SAST linter with custom rules, CI/CD security pipeline, security scanning dashboard
- Labs: WebSockets, Web cache poisoning (15 labs, total: 111/211)

Phase 5: Research & Discovery (Weeks 14-15, 48 hours)

- Week 14: API Security Scanner Development (24h)
- Week 15: PyJWT Security Audit & CVE Research (24h)
- Security Study: Hacking APIs Ch 5-7, 13-14, OWASP API Security Top 10
- Deliverable: API scanner (REST + GraphQL), PyJWT security audit published, CVE methodology
- Labs: API testing, JWT attacks (10 labs, total: 121/211)

Phase 6: Enterprise Production Tools (Weeks 16-18, 72 hours)

- Week 16: Enterprise Security Framework Design (24h)
- Week 17: Vulnerability Aggregator (multi-tool integration) (24h)
- Week 18: Dockerization + Production Documentation (24h)
- Security Study: Secure by Design Ch 7-10, Full Stack Python Security Ch 7-9
- Deliverable: Enterprise security framework, vulnerability aggregator, all tools Dockerized
- Labs: XXE, Insecure deserialization, Server-side template injection (20 labs, total: 141/211)

Phase 7: Portfolio & System Design (Week 19, 24 hours)

- Portfolio website development with all tools showcased
- P2P project public launch preparation (15+ exercises)
- AppSec System Design interview practice (STRIDE methodology)
- Security Study: OWASP ASVS, AppSec interview question repositories
- Deliverable: Portfolio website live, P2P project public, interview-ready system design skills

Phase 8: Cloud Security Deep Dive (Weeks 20-21, 48 hours)

- Week 20: AWS Security (IAM, S3, Lambda, API Gateway) (24h)
- Week 21: Kubernetes Security (RBAC, network policies, pod security) + IaC (24h)
- Security Study: AWS/K8s documentation, OWASP Kubernetes Cheat Sheet, Terraform security
- Deliverable: AWS security scanner, Kubernetes RBAC auditor, IaC vulnerability scanner
- Labs: flAWS (6 levels), CloudGoat scenarios, Kubernetes Goat

Phase 9: DevSecOps & Modern APIs (Weeks 22-23, 48 hours)

- Week 22: CI/CD Security Gates + HashiCorp Vault (24h)
- Week 23: GraphQL Security + Service Mesh (Istio) (24h)
- Security Study: GraphQL Cheat Sheet, Vault documentation, Istio security
- Deliverable: GraphQL security scanner, CI/CD security gates, Vault integration, service mesh basics
- Labs: GraphQL vulnerabilities (10 labs, total: 151/211)

Phase 10: Final Preparation (Weeks 24-28, 115 hours)

- Week 24: Async Python (asyncio, aiohttp) + Distributed Systems (23h)
- Week 25: Tool Polish + Documentation Completion (23h)
- Week 26: Advanced Interview Prep + Mock Interviews (23h)
- Week 27: Active Job Applications (23h)
- Week 28: Networking + Final Portfolio Updates (23h)
- Security Study: Python asyncio docs, OWASP Secure Coding Practices, Interview prep
- Deliverable: Async security scanner, interview-ready portfolio, active job applications

28-WEEK SUCCESS METRICS

Technical Skills Mastery - Elite Level:

- Python Fluency: 8.5/10 level - All 12 Python Workout chapters (50+ exercises), async patterns, production coding
- Web Security: Complete OWASP Top 10 mastery, 150+ PortSwigger labs, Burp Suite expert proficiency
- SQL Injection: Expert level with 40 dedicated labs (basic, UNION, blind, second-order, database-specific)
- XSS Mastery: All contexts (reflected, stored, DOM-based), CSP bypasses, context-aware encoding
- CSRF/SSRF: Token bypass techniques, SameSite bypasses, cloud metadata exploitation
- Authentication: OAuth 2.0 implementation, JWT security, session management, MFA, RBAC/ABAC
- Authorization: IDOR detection, BOLA/BFLA exploitation, privilege escalation testing
- API Security: REST + GraphQL + gRPC expertise, microservices patterns, OWASP API Top 10
- Cloud Security: AWS (IAM, S3, Lambda), Kubernetes (pods, RBAC, network policies, secrets), IaC scanning
- DevSecOps: CI/CD security (GitHub Actions, GitLab CI), Vault, supply chain security, Docker security
- Security Automation: SAST/DAST tools (Semgrep, Bandit, OWASP ZAP), custom rules, security dashboards
- Security Architecture: STRIDE threat modeling, OWASP ASVS vocabulary, Secure-by-Design principles
- Advanced Python: Async/await, aiohttp, production tooling, platform integrations, OOP mastery

Deliverables & Portfolio - Production Quality:

- Security Tools: 20+ production-ready tools (SQLi scanner, blind SQLi exploiter, XSS payload generator, CSRF PoC generator, SSRF scanner, IDOR detector, authorization testing toolkit, OAuth 2.0 server, auth system with MFA, SAST dashboard, API scanner, vulnerability aggregator, AWS security scanner, Kubernetes RBAC auditor, GraphQL scanner, async security scanner)
- P2P Project: 20+ comprehensive secure coding exercises with ~30 tests each, professional documentation, public GitHub presence with star growth strategy
- Blog Posts: 8-10 technical articles on dev.to establishing public presence and thought leadership
- Hands-On Labs: 150+ PortSwigger labs, flAWS (6 levels), CloudGoat, Kubernetes Goat, DVGA
- Research: PyJWT security audit published, CVE discovery methodology mastered, potential 1-2 CVEs filed
- Portfolio Website: Professional showcase of all tools, blog posts, and technical capabilities

Interview Readiness - Top 5-10% Candidate:

- Technical Questions: Can answer 80-90% of AppSec interview questions covering web security, cloud security, DevSecOps, modern APIs, threat modeling
- Hands-On Assessments: Can complete live coding for vulnerability detection, security tool building, secure code review with confidence
- System Design: Can architect secure systems using STRIDE, explain defense-in-depth, design threat models, apply Secure-by-Design principles
- Behavioral Stories: 12+ STAR-format stories from Intel demonstrating technical leadership and security impact
- Unique Differentiators: Intel enterprise experience (553+ threats documented) + modern cloud/DevSecOps skills + P2P open source leadership + comprehensive blog presence

CRITICAL RESOURCES

Must-Have Bookmarks:

- OWASP Top 10: https://owasp.org/www-project-top-ten/
- OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/
- PortSwigger Web Security Academy: https://portswigger.net/web-security
- OWASP API Security Top 10: https://owasp.org/API-Security/
- OWASP Testing Guide: https://owasp.org/www-project-web-security-testing-guide/
- OWASP ASVS: https://owasp.org/www-project-application-security-verification-standard/

Your Books (Already Owned):

- Python Workout (2nd Edition) - Reuven M. Lerner
- Effective Python (3rd Edition) - Brett Slatkin
- API Security in Action - Neil Madden (Manning, 2020)
- Secure by Design - Dan Bergh Johnsson, Daniel Deogun, Daniel Sawano (Manning, 2019)
- Full Stack Python Security - Dennis Byrne (Manning, 2021)
- Hacking APIs - Corey J. Ball (No Starch Press, 2022)

Weekly Schedule Template:

- Monday-Friday (Weekdays): Morning (2h) book reading, Afternoon (3h) hands-on practice, Evening (1-2h) labs or writing
- Saturday: Morning-Afternoon (5-6h) major project work, Evening (2h) community engagement
- Sunday: Morning (2-3h) review and planning, Afternoon: rest and recovery
- Total: 24 hours per week (sustainable long-term pace)

IMPORTANT NOTES & MERGER HIGHLIGHTS

COMPREHENSIVE MERGER: This 28-week curriculum preserves 100% of content from both the 21-week intensive plan and the 26-week sustainable plan. No content has been cut or reduced.

SQL INJECTION MASTERY: Maintains the deep 3-week SQL injection coverage from the 21-week plan (Weeks 1, 2, 4) with 40 dedicated labs - far more comprehensive than typical bootcamp coverage.

SUSTAINABLE PACE: Adopts the 24 hrs/week average from the 26-week plan (vs 27 hrs/week in 21-week plan) for better work-life balance while maintaining intensive learning.

EXTENDED TIMELINE: The 28-week duration (vs 21 or 26 weeks) allows for comprehensive coverage without rushing, better knowledge retention, and more time for portfolio development.

BURP SUITE INTEGRATION: Week 3 adds dedicated Burp Suite mastery not present in the 26-week plan, critical for professional AppSec work.

SECURITY STUDY HOURS: Integrates the detailed 8-hour security study blocks from the 26-week plan for deeper theoretical understanding.

ALL URLS FREE: All URLs are free and publicly accessible unless specified. Book chapters reference your owned physical books.

READING ORDER: Complete Security Study readings BEFORE attempting exercises for best learning outcomes. Take detailed notes on key concepts for later reference during interviews.

DEFENSIVE FOCUS: ALWAYS prioritize 'How to Prevent' sections in readings for defensive knowledge - this differentiates you from pure pentesters.

PORTFOLIO VISIBILITY: Maintain daily GitHub commit habit to build portfolio visibility. Aim for green squares every day across the 28 weeks.

DEPTH OVER BREADTH: Focus on understanding WHY vulnerabilities exist and HOW to prevent them, not just exploitation. This curriculum transforms you from Intel security engineer to complete AppSec engineer.

WEEK 1 CRITICAL: Week 1 Security Concepts reading is CRITICAL - teaches defensive patterns needed for all exercises. Don't skip the prevention sections.

INTEL ADVANTAGE: Leverage your unique Intel threat modeling experience (553+ threats documented) as a differentiator - most AppSec candidates lack this systematic approach.

C/SYSTEMS BACKGROUND: Your C/systems programming experience provides advantages in understanding memory management, performance implications, and low-level security concepts that most web-focused AppSec candidates lack.

Timeline & Targets:

- Start Date: December 2, 2025 (Week 1)
- End Date: June 15, 2026 (Week 28)
- Job Search: Begin Week 24, intensify Weeks 27-28
- Target Start: July 2026

This comprehensive 28-week curriculum is your complete roadmap to becoming a competitive AppSec Engineer candidate. It preserves every element from both the intensive 21-week plan and the sustainable 26-week plan, ensuring you master SQL injection at expert level, develop production-quality security tools,

establish public presence through blogging, build a robust portfolio, and prepare thoroughly for technical interviews. The extended timeline allows for deep learning, better retention, and a more sustainable pace that prevents burnout while maintaining the intensity needed for professional excellence.