Complete 28-Week FULL-TIME INTENSIVE Application Security Engineering Curriculum

ALL WEEKS COMPREHENSIVE EDITION - 40 Hours/Week

Duration:	28 Weeks (Dec 2, 2025 - Jun 14, 2026)
Weekly Hours:	40 hours (8 hrs/day Mon-Fri)
Total:	1,120 hours intensive study
Job Search:	ACTIVE - 3-4 applications daily since Week 1
Target:	Remote AppSec Engineer ($125K-$145K)

Generated: December 07, 2025

WEEK 1: Python Fundamentals & SQL Injection Foundation

40 Hours Total

Python Workout Ch 1-2 COMPLETE (8 hours)

All exercises: number guessing, summing, running average, hexadecimal, mysum, run_timing, word statistics

Effective Python Items 1-10 COMPLETE (4 hours - REQUIRED)

Python version, PEP 8, bytes vs str, f-strings, helper functions

Security Study (12 hours) - EXPLICIT READINGS:

1. OWASP Top 10 2021 - Complete (4 hours) | https://owasp.org/Top10/
2. PortSwigger SQL Injection Complete Guide (4 hours) | https://portswigger.net/web-security/sql-injection
3. OWASP Testing Guide - Input Validation (2 hours)
4. OWASP SQL Injection Prevention Cheat Sheet (2 hours)

Practice (8 hours):

Build 3 tools: (1) Advanced password validator with entropy, (2) SQL injection pattern recognizer (50+ patterns), (3) Vulnerable Flask app for SQLi testing

Labs (6 hours):

SQLBolt complete (18 lessons) + PortSwigger SQLi labs (15 labs: Apprentice + half Practitioner, total: 15/211)

Writing (2 hours):

Blog: 'SQL Injection in 2025: Why Parameterized Queries Aren't Enough' (1500+ words)
Deliverable: 3 tools, 15 labs (15/211), blog post, Python 6/10

WEEK 2: Strings, Lists & Advanced SQL Injection

40 Hours Total

Python Workout Ch 3-4 COMPLETE (8 hours)

All exercises: pig Latin, palindrome, sorting, summing, averaging, unique elements, alphabetizing

Effective Python Items 11-20 COMPLETE (4 hours - REQUIRED)

Slicing, unpacking, enumerate, zip, avoid else after loops

Security Study (10 hours) - EXPLICIT READINGS:

1. Secure by Design Ch 1-3 COMPLETE (5 hours) | Pages 3-76
2. API Security in Action Ch 2 (3 hours) | Pages 33-68, prepared statements
3. OWASP Input Validation Cheat Sheet (2 hours)

Practice (10 hours):

Build 3 tools: (1) SQLi code review scanner (AST-based), (2) JWT security analyzer, (3) Input validation library (OWASP)

Labs (6 hours):

PortSwigger SQLi advanced (15 labs including UNION attacks, total: 30/211)

Writing (2 hours):

Blog: 'Building Secure-by-Design: Intel Threat Modeling at Scale' (1500+ words)
Deliverable: 6 tools total, 30 labs (30/211), 2 blogs, Python 7/10

WEEK 3: Dictionaries, Sets & Burp Suite Mastery

40 Hours Total

Python Workout Ch 5 COMPLETE (8 hours)

All dict/set exercises: restaurant orders, word counts, flip dictionary, unique vowels

Effective Python Items 21-30 COMPLETE (4 hours - REQUIRED)

Dictionary patterns, get vs keys, defaultdict, comprehensions mastery

Security Study (12 hours) - EXPLICIT READINGS:

1. API Security in Action Ch 1-3 (6 hours) | Full chapters with examples
2. Burp Suite Complete Documentation (4 hours) | Getting Started + Scanner tutorials
3. Burp Suite Practical Practice (2 hours hands-on)

Practice (8 hours):

Build advanced SQLi detector with dict/set deduplication, database fingerprinting, Burp Suite extension

Labs (6 hours):

Burp intensive (4 hrs) + PortSwigger XSS (15 labs: reflected/stored/DOM, total: 45/211)

Writing (2 hours):

Blog: 'From Intel Cryptography to Python AppSec' (1500+ words)
Deliverable: 7 tools + Burp extension, 45 labs, 3 blogs, Burp proficient, Python 7.5/10

WEEK 4: Files, Context Managers & Blind SQL Injection Mastery

40 Hours Total

Python Workout Ch 6 COMPLETE (8 hours)

All file ops: reading/writing, CSV/JSON parsing, log analysis

Effective Python Items 31-40 COMPLETE (4 hours - REQUIRED)

File handling, context managers, with statement mastery

Security Study (12 hours) - EXPLICIT READINGS:

1. Hacking APIs Ch 1-3 (6 hours) | API fundamentals + reconnaissance
2. PortSwigger Blind SQLi MASTERY (4 hours) | Boolean/time/error/OAST techniques
3. PortSwigger SQLi Cheat Sheet MASTERY (2 hours) | All databases

Practice (10 hours):

Build production blind SQLi framework: Boolean exploiter, time-based detector, file-based payloads, automated data exfiltration

Labs (4 hours):

PortSwigger blind SQLi intensive (15 labs including OAST, total: 60/211)

Writing (2 hours):

Blog: 'Blind SQL Injection: Boolean to OAST Techniques' (2000+ words with tool demo)
Deliverable: Advanced blind SQLi framework, 60 labs, 4 blogs, SQL MASTERY, Python 8/10

WEEK 5: Functions, Decorators & XSS Deep Dive

40 Hours Total

Python Workout Ch 7 COMPLETE (8 hours)

Functions: parameters, *args, **kwargs, closures, decorators for logging/timing

Effective Python Items 41-50 COMPLETE (4 hours - REQUIRED)

Function patterns, closures, decorators

Security Study (10 hours) - EXPLICIT READINGS:

1. PortSwigger XSS Complete Guide (4 hours) | Reflected/Stored/DOM, all contexts
2. PortSwigger CSP (2 hours) | CSP bypasses, unsafe-inline
3. OWASP XSS Prevention Cheat Sheet (2 hours)
4. OWASP DOM XSS Prevention (2 hours)

Practice (10 hours):

Build XSS payload generator with modular functions, decorators for encoding (HTML/URL/JS), context-aware selection

Labs (6 hours):

PortSwigger advanced XSS (25 labs: all contexts + CSP bypasses, total: 85/211)

Writing (2 hours):

Blog: 'XSS Mastery: From Context-Aware Encoding to CSP Bypasses' (1800+ words)
Deliverable: XSS tool with decorators, 85 labs, 5 blogs, Python 8/10

WEEK 6: CSRF, SSRF & Server-Side Attacks

40 Hours Total

Python Review (4 hours)

Review Ch 1-7, refactor tools using comprehensions

Effective Python Items 51-60 COMPLETE (4 hours - REQUIRED)

Code organization, modules for namespaces

Security Study (12 hours) - EXPLICIT READINGS:

1. PortSwigger CSRF Complete (4 hours) | Token bypass, SameSite exploitation
2. PortSwigger SSRF Complete (4 hours) | Blind SSRF, cloud metadata (169.254.169.254)
3. API Security in Action Ch 10 (3 hours) | Microservices SSRF, DNS rebinding
4. OWASP SSRF Prevention (1 hour)

Practice (10 hours):

Build CSRF PoC generator with comprehensions for form extraction, SameSite bypass payloads, clickjacking iframes

Labs (8 hours):

PortSwigger CSRF (10 labs) + SSRF (10 labs, total: 105/211)

Writing (2 hours):

Blog: 'CSRF in 2025: SameSite Bypasses & Modern Attack Vectors' (1600+ words)
Deliverable: CSRF/SSRF tools, 105 labs, 6 blogs, Python 8.5/10

WEEK 7: Access Control, IDOR & Business Logic

40 Hours Total

Python Review (4 hours)

Revisit challenging exercises Ch 1-7

List Comprehensions Practice (4 hours - REQUIRED)

Advanced list/dict/set comprehensions for security data transformation

Security Study (12 hours) - EXPLICIT READINGS:

1. PortSwigger Access Control Complete (5 hours) | Vertical/horizontal, IDOR, parameter-based
2. PortSwigger Business Logic (3 hours) | Logic flaw patterns
3. Hacking APIs Ch 10 (3 hours) | BOLA/BFLA, A-B testing | Pages 200-230
4. OWASP IDOR Prevention (1 hour)

Practice (10 hours):

Build authorization toolkit: IDOR detector (fuzzes IDs), privilege escalation checker, A-B testing logic, modular Python Workout patterns

Labs (8 hours):

PortSwigger access control (15 labs) + business logic (10 labs, total: 130/211)

Writing (2 hours):

Blog: '5 Authorization Bugs AI Code Generation Misses' (1500+ words)
Deliverable: Authorization toolkit, 130 labs, 7 blogs, Phase 2 complete

WEEK 8: Modules, Packages & OAuth 2.0 Deep Dive

40 Hours Total

Python Workout Ch 8 COMPLETE (6 hours)

Sales tax module, menu package system, module organization patterns

Effective Python Items 61-70 COMPLETE (4 hours - REQUIRED)

Generator expressions, yield from, itertools

Security Study (12 hours) - EXPLICIT READINGS:

1. API Security in Action Ch 6-7 (6 hours) | JWT claims/JOSE, OAuth2 flows, PKCE | Pages 150-220
2. PortSwigger OAuth Vulnerabilities (3 hours) | Authorization code injection, token leakage
3. Full Stack Python Security Ch 11 (3 hours) | Django OAuth | Pages 220-250

Practice (10 hours):

Build OAuth analyzer package: token inspection (JWT decode/validation), misconfiguration detection (missing state/PKCE), assessment report

Labs (6 hours):

PortSwigger OAuth (10 labs) + JWT (10 labs, total: 150/211)

Writing (2 hours):

Blog: 'OAuth 2.0 Security: From Authorization Code to PKCE Bypasses' (1700+ words)
Deliverable: OAuth analyzer, JWT decoder, 150 labs, 8 blogs

WEEK 9: OOP Part 1 & Complete Authentication System

40 Hours Total

Python Workout Ch 9 (6 hours)

Ice cream classes, bowl limits, inheritance patterns

Effective Python Items 71-80 COMPLETE (4 hours - REQUIRED)

Compose classes, @classmethod, super(), mix-ins

Security Study (12 hours) - EXPLICIT READINGS:

1. Full Stack Python Security Ch 7-9 (6 hours) | Sessions/Auth/Passwords | Pages 140-200
2. API Security in Action Ch 4 (3 hours) | Session cookies, CSRF, SameSite | Pages 90-120
3. Hacking APIs Ch 8 (3 hours) | Attacking authentication, JWT None attack | Pages 160-190

Practice (10 hours):

Build auth module (OOP): User class (Argon2id hashing), Session class (secure tokens), AuthManager (lifecycle), rate limiting decorator

Labs (6 hours):

PortSwigger Authentication advanced (10 labs) + Business Logic (5 labs, total: 165/211)

Writing (2 hours):

Blog: 'Building Secure Authentication: Beyond Bcrypt' (1600+ words)
Deliverable: Auth module OOP, 165 labs, 9 blogs

WEEK 10: OOP Part 2 & P2P Exercise Creation

40 Hours Total

Python Workout Ch 9 continued (6 hours)

FlexibleDict, animals hierarchy, cages composition, zoo system

Effective Python Items 81-90 COMPLETE (4 hours - REQUIRED)

Public attributes, collections.abc inheritance, @property

Security Study (10 hours) - EXPLICIT READINGS:

1. API Security in Action Ch 8-9 (5 hours) | RBAC/ABAC, macaroons | Pages 220-280
2. Full Stack Python Security Ch 10 (3 hours) | Authorization, permissions | Pages 200-220
3. OWASP Access Control Cheat Sheet (2 hours)

Practice (6 hours):

Build RBAC/ABAC framework (OOP): Policy hierarchy, enforcement decorators, audit trail observer pattern

Writing (2 hours):

Blog: 'Common RBAC Implementation Mistakes' (1500+ words)

P2P Project (8 hours):

Create 7 P2P auth/authz exercises with 30+ tests each: password hashing, JWT validation, session mgmt, RBAC, OAuth flow
Deliverable: RBAC framework, 7 P2P exercises (210+ tests), 165 labs maintained, 10 blogs

WEEK 11: Iterators, Generators & SAST Tools

40 Hours Total

Python Workout Ch 10 COMPLETE (6 hours)

MyEnumerate, circle iterator, all lines generator, elapsed since, MyChain

Effective Python Review (4 hours - REQUIRED)

Review all 90 items systematically for interview prep

Security Study (12 hours) - EXPLICIT READINGS:

1. Semgrep Documentation - Rules (5 hours) | https://semgrep.dev/docs/ | Create 15+ custom rules
2. Bandit Documentation (3 hours) | https://bandit.readthedocs.io/ | AST-based detection
3. OWASP Code Review Guide (2 hours) | Taint analysis, data flow
4. SonarQube Community (2 hours) | Quality gates, Python plugin

Practice (10 hours):

Build SAST orchestrator (generators): iterate files yielding findings, parallel scanning with yield from, custom Semgrep rules, multi-tool aggregation

Labs (6 hours):

OWASP Juice Shop (20 challenges: injection/XSS/auth) + PortSwigger WebSockets (5 labs, total: 170/211)

Writing (2 hours):

Blog: 'Building Production SAST: From Semgrep to Custom Rules' (1800+ words)
Deliverable: SAST orchestrator, 15 Semgrep rules, 170 labs, 11 blogs

WEEK 12: Async Foundations & CI/CD Security

40 Hours Total

Python Async Introduction (6 hours)

asyncio basics, async/await patterns, concurrent I/O

Python Workout Review (4 hours)

Review all chapters, identify async-applicable patterns

Security Study (12 hours) - EXPLICIT READINGS:

1. GitHub Actions Security Hardening (5 hours) | GITHUB_TOKEN, secrets, runners
2. GitLab CI/CD Security (3 hours) | Protected variables, job tokens, scanning
3. Pre-commit Framework (2 hours) | Hook config, security integration
4. OWASP DevSecOps Guideline (2 hours) | Pipeline gates, shift-left

Practice (10 hours):

Build CI/CD scanner: analyzes GitHub Actions/GitLab YAML, detects insecure patterns (hardcoded secrets, overly permissive tokens), remediation report

Labs (6 hours):

CI/CD security implementation: GitHub workflow with Semgrep/Bandit/Gitleaks, pre-commit hooks, PR gates + PortSwigger Cache Poisoning (5 labs, total: 175/211)

Writing (2 hours):

Blog: 'Shift-Left Security: Implementing Pre-Commit Hooks at Scale' (1600+ words)
Deliverable: CI/CD scanner, secure workflows, pre-commit config, 175 labs, 12 blogs

WEEK 13: Security Framework Dashboard & Tool Integration

40 Hours Total

Python OOP Review (6 hours)

Consolidate OOP and generator patterns from Weeks 7-12

Advanced Python Patterns (4 hours)

Concurrent programming patterns, ThreadPoolExecutor basics

Security Study (10 hours) - EXPLICIT READINGS:

1. OWASP ZAP API Scanning (4 hours) | Automation, OpenAPI import, CI/CD integration
2. Nuclei Template Documentation (3 hours) | Template syntax, severity, workflow chaining
3. Hacking APIs Ch 4 (3 hours) | Tool ecosystem, Postman automation | Pages 80-110

Practice (12 hours):

Build security dashboard: SAST aggregation (Semgrep/Bandit), DAST (ZAP API), dependency vuln (Safety), secret scanning (Gitleaks), HTML dashboard with trends

Labs (6 hours):

PortSwigger HTTP Request Smuggling (10 labs, total: 185/211)

Writing (2 hours):

Blog: 'Building a Python Security Dashboard: 5 Tools in 200 Lines' (1700+ words)
Deliverable: Security dashboard v1.0, 185 labs, 13 blogs, Phase 4 complete

WEEK 14: Advanced Iteration & API Security Scanner

40 Hours Total

Advanced Itertools (6 hours)

Master itertools module, custom security-focused iterators for fuzzing

Python Performance (4 hours)

Profile before optimizing, decimal for precision

Security Study (12 hours) - EXPLICIT READINGS:

1. Hacking APIs Ch 5-7 (6 hours) | Discovery/recon, Kiterunner, endpoint analysis | Pages 110-160
2. OWASP API Security Top 10 - Full Review (4 hours) | API1-API10 with detection
3. PortSwigger API Testing Guide (2 hours) | Documentation exploitation, hidden endpoints

Practice (10 hours):

Build API scanner with itertools: endpoint discovery (itertools.product for paths), parameter fuzzing (chain/cycle), auth bypass (permutations), rate limit detection

Labs (6 hours):

PortSwigger API Testing (10 labs) + Business Logic (5 labs, total: 200/211)

Writing (2 hours):

Blog: 'API Security Scanner Architecture: From Discovery to Exploitation' (1900+ words)
Deliverable: API scanner v1.0, 200 labs, 14 blogs

WEEK 15: PyJWT Security Audit & CVE Research Methodology

40 Hours Total

Python Security Review (6 hours)

Review all security tools, consolidate async patterns

Advanced Data Structures (4 hours)

deque, bisect, heapq, memoryview

Security Study (10 hours) - EXPLICIT READINGS:

1. Hacking APIs Ch 13-14 (5 hours) | Evasion, rate limit bypass, GraphQL | Pages 280-330

2. PortSwigger JWT Attacks Complete (3 hours) | Algorithm confusion, key injection, JKU/X5U
3. Critical JWT CVEs Research (2 hours) | CVE-2015-9235 (None), CVE-2016-10555 (confusion)

Practice (12 hours):

PyJWT security audit: clone repo, review crypto implementation, test algorithm confusion, check defaults, document findings, responsible disclosure prep

Writing (2 hours):

Blog: 'PyJWT Security Audit: A Junior Engineer's Deep Dive' (2000+ words)

Deliverable: PyJWT audit report, JWT attack tool, 200 labs, 15 blogs, Phase 5 complete

WEEK 16: Enterprise Security Framework Design

40 Hours Total

Design Patterns (6 hours)

Factory, Strategy, Observer patterns for security tools

Testing Best Practices (4 hours)

TestCase, setUp/tearDown, mocks, dependency encapsulation

Security Study (12 hours) - EXPLICIT READINGS:

1. Secure by Design Ch 7-8 (6 hours) | State security, pipeline security, deep modeling | Pages 140-200
2. Full Stack Python Security Ch 4-6 (4 hours) | Cryptography: AES/RSA/TLS | Pages 70-140
3. OWASP ASVS v5.0 - Architecture (2 hours) | V1, V10, V14 requirements

Practice (10 hours):

Design enterprise framework: Plugin system (Factory), scanning modes (Strategy), real-time alerts (Observer), UML diagrams, API specs, vulnerability data models

Labs (6 hours):

PortSwigger Insecure Deserialization (6 labs) + Prototype Pollution (5 labs, total: 211/211 COMPLETE!)

Writing (2 hours):

Blog: 'Enterprise AppSec Architecture: Design Patterns That Scale' (1800+ words)

Deliverable: Framework architecture doc, 211 labs COMPLETE, 16 blogs

WEEK 17: Vulnerability Aggregator Implementation

40 Hours Total

Framework Implementation (8 hours)

Build core framework implementing designed architecture

Python Packaging (4 hours)

Community modules, virtual envs, docstrings, packages

Security Study (10 hours) - EXPLICIT READINGS:

1. Secure by Design Ch 9-10 (5 hours) | Exception handling, cloud security, 12-factor
2. Snyk API Integration (2 hours) | REST API, webhooks | https://docs.snyk.io/
3. GitLeaks & TruffleHog (3 hours) | CI/CD integration, output formats

Practice (10 hours):

Build aggregator: unified interface (Semgrep/Bandit/Safety/Gitleaks), deduplication logic, severity normalization (CVSS), trend analysis, REST API, SQLite backend

Writing (2 hours):

Blog: 'Building a Vulnerability Aggregator: From Chaos to Clarity' (1700+ words)

Deliverable: Vulnerability aggregator v1.0, REST API docs, 17 blogs, 211 labs maintained

WEEK 18: Dockerization & Production Documentation

40 Hours Total

Python Packaging Final (6 hours)

setup.py/pyproject.toml, entry points, distribution-ready

Final Python Review (4 hours)

Root exceptions, break circular deps, warnings, typing

Security Study (10 hours) - EXPLICIT READINGS:

1. Docker Security Best Practices (4 hours) | Multi-stage builds, non-root users, minimal images
2. Trivy Container Scanning (3 hours) | Image scanning, K8s integration
3. Full Stack Python Security Ch 12 (3 hours) | OS security, env vars, secrets | Pages 250-275

Practice (12 hours):

Dockerize framework: multi-stage Dockerfile (security hardened), Docker Compose (full stack), health checks, Docker secrets, Trivy scanning in CI/CD, comprehensive docs (README/API/contributing)

Writing (2 hours):

Blog: 'From Script to Production: Dockerizing Security Tools' (1800+ words)

Deliverable: Dockerized framework, complete docs, PyPI-ready, 18 blogs, Phase 6 complete

WEEK 19: Portfolio Website & AppSec System Design

40 Hours Total

Writing (2 hours):

Blog: 'AppSec System Design: Thinking Like an Architect' (2000+ words)

P2P Project (6 hours):

Finalize P2P exercises (comprehensive tests), onboarding docs, GitHub org setup, OWASP LA presentation prep, marketing materials

Portfolio Development (12 hours):

Create GitHub Pages portfolio: tool demos (screenshots/GIFs), lab statistics, blog collection, professional bio, resume integration

System Design Practice (10 hours):

Practice 5 scenarios: (1) Secure auth for microservices (JWT/SSO/gateway), (2) Multi-tenant SaaS API (BOLA prevention), (3) CI/CD secrets (Vault), (4) Zero-trust network, (5) Threat model e-commerce platform. Document with STRIDE, sequence diagrams, trade-offs

Intel Experience Documentation (4 hours):

Translate Intel TMT to system design language, document scalability achievements, prepare STAR stories

Interview Preparation (6 hours):

Create 1-page Security Design Portfolio, compile AppSec Q&A; bank, practice explaining concepts, review OWASP ASVS

Deliverable: Portfolio live, system design portfolio, interview-ready, P2P launch-ready, 19 blogs

WEEK 20: AWS Security Mastery & IaC Scanning

40 Hours Total

Security Study (14 hours) - EXPLICIT READINGS:

1. AWS IAM Best Practices (4 hours) | Least privilege, MFA, Access Analyzer | https://docs.aws.amazon.com/IAM/
2. AWS S3 Security (3 hours) | Block Public Access, bucket policies, encryption
3. AWS Lambda Security (3 hours) | Function URL auth, VPC, secrets management
4. AWS API Gateway (2 hours) | Lambda authorizers, Cognito, WAF rules
5. Checkov IaC Scanning (2 hours) | Terraform/CloudFormation rules | https://www.checkov.io/

Practice (10 hours):

Build AWS scanner (boto3): IAM policy analyzer (overly permissive), S3 config audit (public/encryption), Lambda security review, Terraform/CloudFormation scanning, CIS AWS Benchmark compliance report

Labs (6 hours):

AWS Free Tier security exercises, Terraform security lab (fix vulnerabilities), LocalStack AWS API simulation

Writing (2 hours):

Blog: 'AWS Security Automation: From IAM to Infrastructure-as-Code' (1900+ words)

Cloud Security Fundamentals (4 hours):

Review shared responsibility model, cloud-native security principles

Deliverable: AWS security scanner, IaC templates, cloud fundamentals, 20 blogs

WEEK 21: Kubernetes Security & CloudGoat Exploitation

40 Hours Total

Security Study (14 hours) - EXPLICIT READINGS:

1. Kubernetes RBAC Documentation (4 hours) | Roles, ClusterRoles, service accounts | https://kubernetes.io/docs/
2. Kubernetes Network Policies (3 hours) | Ingress/egress rules, namespace isolation, CNI
3. Pod Security Standards (3 hours) | Privileged/Baseline/Restricted profiles, Pod Security Admission
4. CloudGoat by Rhino Security Labs (2 hours) | Setup, scenarios, attack methodologies | https://github.com/RhinoSecurityLabs/cloudgoat
5. Trivy Kubernetes Scanning (2 hours) | Misconfiguration scanning, CIS benchmarks

Practice (10 hours):

CloudGoat AWS labs: iam_privesc_by_rollback (enumerate IAM), ec2_ssrf (SSRF for metadata), cloud_breach_s3 (S3 exfiltration), codebuild_secrets (plaintext secrets). Document attack paths + remediation

Labs (6 hours):

K8s security: deploy vulnerable pods (fix with Pod Security Standards), implement network policies (microservice isolation), RBAC hardening (multi-tenant), Falco rules (runtime detection)

Writing (2 hours):

Blog: 'Kubernetes RBAC Gone Wrong: From Pod to Cluster Admin' (1800+ words)

Container Security (4 hours):

Review container isolation, namespace security, cgroup limits

Deliverable: CloudGoat completion, K8s security templates, 21 blogs, Phase 8 complete

WEEK 22: Advanced CI/CD Security & HashiCorp Vault

40 Hours Total

Security Study (14 hours) - EXPLICIT READINGS:

1. GitHub Actions Advanced Security (4 hours) | CodeQL custom queries, secret scanning, dependency review
2. GitLab Security Scanning (3 hours) | SAST/DAST/container scanning, license compliance
3. HashiCorp Vault Getting Started (4 hours) | Secret engines, auth methods, policies | https://developer.hashicorp.com/vault/
4. Vault CI/CD Integration (3 hours) | AppRole auth, dynamic secrets, transit encryption

Practice (10 hours):

Build advanced CI/CD pipeline: GitHub Actions with security gates (SAST/secrets scan required), Vault integration (runtime secrets), dynamic DB credentials (Vault), policy-as-code (OPA/Conftest), automated security reporting (Slack/Teams)

Labs (6 hours):

Vault tutorials (secrets engines, PKI, transit), GitHub Advanced Security trial exercises, security gate blocking PRs with critical findings

Writing (2 hours):

Blog: 'Zero-Trust CI/CD: From Hardcoded Secrets to Vault' (1900+ words)

DevSecOps Foundations (4 hours):

Review shift-left security, security champions model

Deliverable: Production CI/CD pipeline, Vault patterns, 22 blogs

WEEK 23: GraphQL Security & Service Mesh

40 Hours Total

Security Study (14 hours) - EXPLICIT READINGS:

1. PortSwigger GraphQL API Vulnerabilities (4 hours) | Introspection, batching, injection through variables
2. OWASP GraphQL Cheat Sheet (3 hours) | Query depth limiting, complexity scoring, resolver authorization
3. Hacking APIs Ch 14 (3 hours) | Attacking GraphQL, InQL Burp extension, mutations | Pages 310-340
4. Istio Security Documentation (3 hours) | mTLS, authorization policies, peer authentication | https://istio.io/
5. OWASP API Security - Modern Patterns (1 hour)

Practice (10 hours):

Build GraphQL testing suite: introspection analyzer (schema extraction), query complexity calculator, batching attack detector, injection fuzzer, authorization bypass tester, integration with API scanner

Labs (6 hours):

PortSwigger GraphQL labs (complete remaining), service mesh security exercises

Writing (2 hours):

Blog: 'GraphQL Security: Beyond Query Depth Limiting' (1800+ words)

Modern API Fundamentals (4 hours):

Review GraphQL vs REST security, service mesh concepts

Deliverable: GraphQL scanner, service mesh checklist, 23 blogs, Phase 9 complete

WEEK 24: Async Python Mastery & Tool Integration

40 Hours Total

Security Study (10 hours) - EXPLICIT READINGS:

1. Distributed Systems Security (3 hours) | CAP theorem security, consensus, Byzantine fault tolerance
2. Modern Authentication (4 hours) | WebAuthn/FIDO2 specs, passkey implementation security
3. AI/ML Security (3 hours) | Prompt injection basics, LLM API security (rate limiting, input validation) | https://portswigger.net/web-security/llm-attacks

Practice (8 hours):

Build async scanner: concurrent port scanning (rate limited), async API endpoint fuzzing, parallel vulnerability verification, results aggregation

Writing (2 hours):

Blog: 'Async Python for Security: 10x Faster Vulnerability Scanning' (1700+ words)

Portfolio Development (4 hours):

Final portfolio updates, LinkedIn with project highlights, resume for target companies

Interview Preparation (6 hours):

Technical practice (OWASP Top 10, API security), system design (3 scenarios from Week 19), behavioral (STAR stories), code review (identify vulns)

Async Python Deep Dive (8 hours):

asyncio fundamentals, aiohttp for async HTTP, concurrent security scanning with rate limiting, async API fuzzing, parallel vuln verification, asyncio.gather aggregation

Deliverable: Async scanner, interview-ready, portfolio finalized, 24 blogs

WEEK 25: Tool Polish & Production Readiness

40 Hours Total

Writing (4 hours):

Blog: 'From Learning to Production: 25 Security Tools in 6 Months' (2500+ words - comprehensive retrospective)

Tool Polish & Documentation (16 hours):

Review all 25+ tools: fix bugs, add error handling, improve UX, consistent style, type hints + mypy, achieve 80%+ test coverage, comprehensive README files, API documentation with examples, architecture decision records (ADRs), video demos for complex tools, contributing guidelines, pin best 6 repos, create GitHub organization, setup GitHub Actions for all repos, add badges (build/coverage/license), compelling descriptions

Deliverable: All tools production-ready, docs complete, GitHub optimized, 25 blogs

WEEK 26: Advanced Interview Preparation

40 Hours Total

Writing (2 hours):

Blog: 'Preparing for AppSec Interviews: What I Learned' (1800+ words)

Advanced Interview Prep (16 hours):

Deep dive into target companies (Trail of Bits, NCC Group, Anthropic, GitLab, Stripe, Coinbase), research recent security incidents and prevention strategies, prepare questions about security culture, review AppSec job descriptions for skill alignment

Mock Interviews (12 hours):

Schedule 5-6 mock interviews with peers/mentors, whiteboard security architecture, timed code review (vuln identification), system design with security focus, practice trade-off explanations verbally

Additional System Design (10 hours):

Additional scenarios: secure file upload (enterprise), fraud detection pipeline (security controls), secure API gateway (partner integrations), DDoS mitigation architecture, secure payment processing system

Deliverable: Mock feedback incorporated, confident in all formats, 26 blogs

WEEK 27: Intensive Job Applications & Networking

40 Hours Total

Writing (2 hours):

Blog: 'The AppSec Job Search: Strategy & Lessons' (1700+ words)

Portfolio Development (8 hours):

Final updates based on feedback, ensure all links work, test all demos, prepare 30-second elevator pitch, prepare 2-minute elevator pitch, update resume with Week 25-26 accomplishments

Active Applications (16 hours):

Apply to 3-4 positions daily (continue active search started Week 1), customize cover letters for each application, track in spreadsheet with follow-up dates, priority targets: Trail of Bits, NCC Group, Anthropic, GitLab, Stripe, Coinbase, additional targets: Yubico, 1Password, Cash App, Wiz, Orca Security, Snyk

Networking (12 hours):

Attend OWASP LA chapter meeting (4th Wednesday), visit Null Space Labs hackerspace (Tuesday nights), connect with AppSec professionals on LinkedIn (50+ connections), reach out to security engineers at target companies, share blog posts in InfoSec communities (Twitter/X, Reddit r/netsec, HackerNews)

Final Portfolio Work (8 hours):

Final updates based on feedback, ensure all links work, test all demos, prepare 30-second elevator pitch, prepare 2-minute elevator pitch, update resume with Week 25-26 accomplishments

Deliverable: Cumulative 400-500+ applications submitted since Week 1, network expanded, 27 blogs

WEEK 28: Final Preparation & Offer Negotiations

40 Hours Total

Writing (2 hours):

Blog: '28 Weeks to AppSec Engineer: The Complete Journey' (3000+ words - final retrospective)

Final Preparation (16 hours):

Review all interview materials, practice explaining complex topics simply, update STAR stories with recent accomplishments, prepare for technical deep dives, review salary negotiation strategies, research compensation data (levels.fyi, Glassdoor), prepare counter-offer strategies, understand equity/benefits packages

Offer Management (10 hours):

Track interview pipelines, manage multiple offer timelines, evaluate offers holistically (comp + culture + growth + WLB), negotiate confidently using data, compare total compensation packages, consider remote work policies and flexibility

Final Polish (10 hours):

Last portfolio updates, ensure all 25+ tools are accessible and documented, final GitHub profile review, update LinkedIn with complete journey, prepare thank-you notes for interviewers, organize references (Jonathan Valamehr, Tony Martin, Brian Nutter from Intel)

Deliverable: Interview-ready, offer negotiation prepared, 28 blogs COMPLETE, 500-700+ total applications, ready to start new role!