Post-28 Week 1-Year Application Security Reading Schedule

COMPREHENSIVE YEAR 1 CONTINUATION

Duration: 52 Weeks (June 16, 2026 - June 14, 2027)

Total Investment: 936 hours across 52 weeks

Average: 18 hours/week (sustainable long-term pace)

Generated: December 7, 2025

Executive Summary

This 52-week Year 1 continuation plan begins immediately after completing the comprehensive 28-week MERGED AppSec curriculum (June 15, 2026). It transforms you from a solid junior AppSec Engineer into a Senior-level technical authority through systematic mastery of advanced offensive security (OSWE), cloud security expertise, custom security tooling, SAST/DAST architecture, and enterprise security architecture.

Goal: Transform from Junior AppSec Engineer to Senior-level technical depth through systematic study of advanced security topics, elite offensive security skills (OSWE), cloud security mastery, DevSecOps automation, custom tooling, and enterprise architecture.

Curriculum Overview by Quarter

Quarter	Weeks	Focus	Weekly Hours	Total Hours
Q1	29-41 (13 weeks)	OSWE + Cloud Security Foundation	22h	286h
Q2	42-54 (13 weeks)	Modern APIs + DevSecOps	18h	234h
Q3	55-67 (13 weeks)	Custom Tooling + SAST/DAST	18h	234h
Q4	68-80 (13 weeks)	Enterprise Architecture + Thought Leadership	14h	182h
Total:				**936 hours across 52 weeks**

Quarter 1: Foundation Building

(Weeks 29-41)

June 16 - September 14, 2026 | 286 Hours Total (22h/week)

Primary Objectives:

- Complete OSWE certification (elite web application exploitation)

- Master AWS security (IAM, S3, Lambda, CloudTrail)
- Achieve Kubernetes security expertise (RBAC, network policies, pod security)
- Discover 2-3 CVEs in cloud-native tools
- Build cloud security assessment toolkit

Week 29: OSWE Introduction + AWS IAM Deep Dive (22 hours)

Books:

- The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 4: Mapping the Application (pp. 151-203)
- AWS Security by Dylan Shield (Manning, 2022), Chapter 6: Managing identities and access (pp. 119-156)

Online Resources:

- OSWE Syllabus & Lab Setup (OffSec official)
- https://www.offsec.com/courses/web-300/
- AWS Well-Architected Security Pillar
- https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/
- OWASP Cloud Security Testing Guide
- https://owasp.org/www-project-cloud-testing/

Practice (12 hours):

- Set up OSWE lab environment
- Build AWS IAM policy analyzer tool (Python)
- Create IAM privilege escalation detection script
- Start documenting OSWE methodology

Labs (5 hours):

- OSWE Initial Labs: Week 1 modules
- AWS IAM CTF challenges

Writing (3 hours):

- Blog Post: "Setting Up Your OSWE Lab: Complete Guide"
- Technical documentation: AWS IAM security patterns

Community (2 hours):

- OWASP LA meeting (4th Wednesday monthly)
- Join OSWE student community forums

Deliverables:

- OSWE lab environment configured
- AWS IAM security analyzer tool
- Blog post published on dev.to

Week 30: Advanced SQL Injection + AWS S3 Security (22 hours)

Books:

- The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 9: Attacking Data Stores (pp. 331-399)
- AWS Security by Dylan Shield (Manning, 2022), Chapter 7: Securing data in S3 (pp. 157-194)

Online Resources:

- PortSwigger Advanced SQLi Labs
- https://portswigger.net/web-security/sql-injection/advanced
- AWS S3 Security Best Practices
- https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-best-practices.html
- OWASP Injection Prevention Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/InjectionPreventionCheatSheet.html

Practice (12 hours):

- OSWE: Advanced SQL injection exploitation techniques
- Build S3 bucket security scanner (misconfigurations, public access)
- Implement second-order SQL injection detector
- Research blind SQLi exploitation in modern frameworks

Labs (5 hours):

- OSWE SQL injection labs
- AWS S3 CTF scenarios (flAWS level 2-3)

Writing (3 hours):

- Blog Post: "Second-Order SQL Injection: Detection and Exploitation"
- Document S3 security misconfigurations

Community (2 hours):

- Null Space Labs Tuesday open night

Deliverables:

- Advanced SQLi exploitation techniques documented
- S3 bucket security scanner tool
- Blog post published

Week 31: XXE & XML Attacks + AWS Lambda Security (22 hours)

Books:

- The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 10: Attacking Back-End Components (pp. 401-476)
- AWS Security by Dylan Shield (Manning, 2022), Chapter 11: Securing serverless applications (pp. 257-292)

Online Resources:

- OWASP XXE Prevention Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/XMLExternalEntityPreventionCheatSheet.html
- AWS Lambda Security Best Practices
- https://docs.aws.amazon.com/lambda/latest/dg/lambda-security.html
- PortSwigger XXE Labs
- https://portswigger.net/web-security/xxe

Practice (12 hours):

- OSWE: XXE exploitation techniques (blind XXE, XXE via file upload)
- Build Lambda function security scanner
- Create XXE payload generator with automated testing
- Research XML deserialization vulnerabilities

Labs (5 hours):

- OSWE XXE labs
- AWS Lambda security challenges

Writing (3 hours):

- Blog Post: "XXE Exploitation in Modern Applications: A Comprehensive Guide"
- Lambda security architecture documentation

Community (2 hours):

- OWASP LA community engagement

Deliverables:

- XXE exploitation toolkit
- Lambda security scanner
- Blog post published

Week 32: Deserialization Attacks + AWS CloudTrail Analysis (22 hours)

Books:

- The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 5: Bypassing Client-Side Controls (pp. 205-262)
- AWS Security by Dylan Shield (Manning, 2022), Chapter 12: Detective controls (pp. 293-330)

Online Resources:

- OWASP Deserialization Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/DeserializationCheatSheet.html
- AWS CloudTrail Security Monitoring
- https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html
- PortSwigger Insecure Deserialization Labs

- https://portswigger.net/web-security/deserialization

Practice (12 hours):

- OSWE: Python/Java deserialization exploitation
- Build CloudTrail anomaly detection system
- Create gadget chain analyzer for popular frameworks
- Research deserialization in modern Python frameworks

Labs (5 hours):

- OSWE deserialization labs
- AWS CloudTrail security analysis exercises

Writing (3 hours):

- Blog Post: "Exploiting Insecure Deserialization in Python Web Applications"
- CloudTrail monitoring best practices documentation

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Deserialization exploitation toolkit
- CloudTrail security monitoring system
- Blog post published

Week 33: Authentication Bypass Techniques + Kubernetes Basics (22 hours)

Books:

- The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 6: Attacking Authentication (pp. 263-300)
- Kubernetes Security by Liz Rice & Michael Hausenblas (O'Reilly, 2021), Chapter 1-2: Kubernetes Security Principles, Pod Security (pp. 1-48)

Online Resources:

- OWASP Authentication Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/AuthenticationCheatSheet.html
- Kubernetes Official Security Documentation
- https://kubernetes.io/docs/concepts/security/
- OWASP Kubernetes Security Testing Guide
- https://github.com/OWASP/www-project-kubernetes-top-ten

Practice (12 hours):

- OSWE: Authentication bypass techniques (timing attacks, race conditions)
- Set up local Kubernetes cluster (minikube/kind)
- Build Kubernetes pod security analyzer

• Research authentication vulnerabilities in microservices

Labs (5 hours):

• OSWE authentication bypass labs

• Kubernetes Goat: Initial levels

Writing (3 hours):

• Blog Post: "Authentication Bypass Techniques: From Theory to Exploitation"

• Kubernetes security fundamentals documentation

Community (2 hours):

• OWASP LA meeting

Deliverables:

• Authentication bypass toolkit

• Kubernetes local environment

• Blog post published

Week 34: Session Management Attacks + Kubernetes RBAC (22 hours)

Books:

• The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 7: Attacking Session Management (pp. 301-330)

• Kubernetes Security by Liz Rice & Michael Hausenblas (O'Reilly, 2021), Chapter 3: Access Control & Authorization (pp. 49-82)

Online Resources:

• OWASP Session Management Cheat Sheet

• https://cheatsheetseries.owasp.org/cheatsheets/SessionManagementCheatSheet.html

• Kubernetes RBAC Documentation

• https://kubernetes.io/docs/reference/access-authn-authz/rbac/

• RBAC Security Best Practices

• https://kubernetes.io/docs/concepts/security/rbac-good-practices/

Practice (12 hours):

• OSWE: Session fixation, prediction, hijacking techniques

• Build Kubernetes RBAC auditor tool

• Create session token analyzer

• Research session management in distributed systems

Labs (5 hours):

• OSWE session management labs

• Kubernetes RBAC privilege escalation scenarios

Writing (3 hours):

- Blog Post: "Session Management Vulnerabilities in Modern Web Applications"
- K8s RBAC security patterns documentation

Community (2 hours):

- Null Space Labs Tuesday engagement

Deliverables:

- Session management testing toolkit
- Kubernetes RBAC auditor
- Blog post published

Week 35: Authorization Flaws + Kubernetes Network Policies (22 hours)

Books:

- The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 8: Attacking Access Controls (pp. 331-382)
- Kubernetes Security by Liz Rice & Michael Hausenblas (O'Reilly, 2021), Chapter 4: Network Security (pp. 83-124)

Online Resources:

- OWASP Authorization Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/AuthorizationCheatSheet.html
- Kubernetes Network Policies
- https://kubernetes.io/docs/concepts/services-networking/network-policies/
- Calico Network Policy Tutorial
- https://docs.tigera.io/calico/latest/network-policy/

Practice (12 hours):

- OSWE: IDOR, privilege escalation, horizontal/vertical access control bypass
- Build Kubernetes network policy analyzer
- Create authorization testing framework
- Research BOLA/BFLA patterns in microservices

Labs (5 hours):

- OSWE access control labs
- Kubernetes network policy exploitation scenarios

Writing (3 hours):

- Blog Post: "Authorization Vulnerabilities: IDOR, BOLA, and BFLA Explained"
- K8s network policy security guide

Community (2 hours):

- OWASP LA engagement

Deliverables:

• Authorization testing framework
• K8s network policy analyzer
• Blog post published

Week 36: Command Injection + K8s Secrets Management (22 hours)

Books:

• The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 10: Attacking Back-End Components (pp. 401-476)
• Kubernetes Security by Liz Rice & Michael Hausenblas (O'Reilly, 2021), Chapter 5: Secrets Management (pp. 125-156)

Online Resources:

• OWASP Command Injection Prevention
• https://cheatsheetseries.owasp.org/cheatsheets/OSCommandInjectionDefenseCheatSheet.html
• Kubernetes Secrets Best Practices
• https://kubernetes.io/docs/concepts/configuration/secret/
• HashiCorp Vault for Kubernetes
• https://developer.hashicorp.com/vault/tutorials/kubernetes

Practice (12 hours):

• OSWE: OS command injection exploitation (blind, time-based)
• Build Kubernetes secrets scanner
• Create command injection payload generator
• Research secrets management in container environments

Labs (5 hours):

• OSWE command injection labs
• Kubernetes Goat secrets challenges

Writing (3 hours):

• Blog Post: "Command Injection: Detection, Exploitation, and Prevention"
• K8s secrets security architecture

Community (2 hours):

• Null Space Labs engagement

Deliverables:

• Command injection toolkit
• K8s secrets security scanner
• Blog post published

Week 37: SSRF Deep Dive + K8s Pod Security Standards (22 hours)

Books:

- • The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 11: Attacking Application Logic (pp. 477-512)
- • Kubernetes Security by Liz Rice & Michael Hausenblas (O'Reilly, 2021), Chapter 2: Pod Security Standards (pp. 25-48)

Online Resources:

- • OWASP SSRF Prevention Cheat Sheet
- • https://cheatsheetseries.owasp.org/cheatsheets/ServerSideRequestForgeryPreventionCheatSheet.html
- • Kubernetes Pod Security Standards
- • https://kubernetes.io/docs/concepts/security/pod-security-standards/
- • Cloud Metadata SSRF Exploitation
- • https://cloud.hacktricks.xyz/pentesting-cloud/aws-security/aws-unauthenticated-enum-access/aws-metadata-ssrf

Practice (12 hours):

- • OSWE: SSRF exploitation (cloud metadata, internal services)
- • Build K8s pod security policy validator
- • Create SSRF payload generator with cloud provider support
- • Research SSRF in containerized environments

Labs (5 hours):

- • OSWE SSRF labs
- • AWS SSRF exploitation scenarios (flAWS level 4-5)

Writing (3 hours):

- • Blog Post: "SSRF in Cloud Environments: Attacking AWS Metadata Service"
- • K8s pod security standards guide

Community (2 hours):

- • OWASP LA meeting

Deliverables:

- • SSRF exploitation toolkit with cloud provider support
- • K8s pod security validator
- • Blog post published

Week 38: File Upload Vulnerabilities + Infrastructure as Code Security (22 hours)

Books:

- • The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 12: Attacking Users (pp. 513-584)
- • Container Security by Liz Rice (O'Reilly, 2020), Chapter 7: Hardening & Securely Distributing (pp. 165-192)

Online Resources:

- OWASP File Upload Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/FileUploadCheatSheet.html
- Terraform Security Best Practices
- https://developer.hashicorp.com/terraform/tutorials/configuration-language/security
- OWASP Infrastructure as Code Security
- https://owasp.org/www-project-devsecops-guideline/latest/02a-Infrastructure-as-Code-Security

Practice (12 hours):

- OSWE: File upload bypass techniques (magic bytes, double extensions)
- Build Terraform/CloudFormation security scanner
- Create file upload vulnerability detector
- Research IaC security scanning tools

Labs (5 hours):

- OSWE file upload labs
- CloudGoat IaC security scenarios

Writing (3 hours):

- Blog Post: "File Upload Vulnerabilities: Bypassing Validation & Exploitation"
- IaC security scanning guide

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- File upload testing toolkit
- IaC vulnerability scanner
- Blog post published

Week 39: Template Injection + Container Image Security (22 hours)

Books:

- The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Chapter 13: Automating Customized Attacks (pp. 585-632)
- Container Security by Liz Rice (O'Reilly, 2020), Chapter 4: Container Images (pp. 81-114)

Online Resources:

- PortSwigger Server-Side Template Injection
- https://portswigger.net/web-security/server-side-template-injection
- OWASP Docker Security Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/DockerSecurityCheatSheet.html
- Container Image Scanning Tools Comparison

- https://docs.docker.com/scout/

Practice (12 hours):

- OSWE: SSTI exploitation (Jinja2, Twig, ERB)
- Build container image vulnerability scanner
- Create template injection payload generator
- Research Dockerfile security best practices

Labs (5 hours):

- OSWE template injection labs
- Container security CTF challenges

Writing (3 hours):

- Blog Post: "Server-Side Template Injection: From Detection to RCE"
- Container image security hardening guide

Community (2 hours):

- OWASP LA engagement

Deliverables:

- Template injection toolkit
- Container image security scanner
- Blog post published

Week 40: OSWE Exam Preparation + Cloud Security Review (22 hours)

Books:

- The Web Application Hacker's Handbook by Stuttard & Pinto (Wiley, 2011), Review key chapters
- Real-World Bug Hunting by Peter Yaworski (No Starch Press, 2019), Selected chapters on methodology (pp. 1-50)

Online Resources:

- OSWE Exam Guide
- https://www.offsec.com/courses/web-300/
- OWASP Testing Guide Review
- https://owasp.org/www-project-web-security-testing-guide/
- AWS & Kubernetes Security Review Checklists
- Multiple vendor resources

Practice (15 hours):

- OSWE practice exams and buffer machines
- Review all exploitation techniques learned
- Build comprehensive exploitation methodology notes
- Practice writing professional penetration test reports

Labs (4 hours):

- Final OSWE lab challenges
- Mock exam scenarios

Writing (2 hours):

- Update portfolio with all Q1 tools
- Prepare OSWE exam strategy document

Community (1 hour):

- Final Q1 OWASP LA meeting

Deliverables:

- OSWE exam-ready methodology
- Complete Q1 portfolio update
- Professional pentest report templates

Week 41: OSWE Exam Week (22 hours)

OSWE Certification Exam:

- 48-hour hands-on exam + 24-hour report

Activities:

- Complete OSWE practical exam (48 hours)
- Write comprehensive penetration test report (24 hours)
- Submit exam report to OffSec

Post-Exam (remaining hours):

- Reflect on exam experience
- Document lessons learned
- Rest and recovery

Deliverables:

- OSWE certification (target)
- Professional pentest report submitted

Quarter 1 Summary: Success Metrics

Completed by September 14, 2026:

- ■ OSWE certification achieved
- ■ 13 production security tools built (AWS IAM analyzer, S3 scanner, Lambda scanner, CloudTrail monitor, K8s RBAC auditor, K8s network policy analyzer, K8s secrets scanner, K8s pod security validator, IaC scanner, container image scanner, and 3+ exploitation toolkits)

- ■ 13 technical blog posts published on dev.to
- ■ AWS security expertise (IAM, S3, Lambda, CloudTrail)
- ■ Kubernetes security mastery (RBAC, network policies, pod security, secrets)
- ■ Infrastructure as Code security scanning capabilities
- ■ Container security expertise
- ■ Advanced exploitation techniques mastered (SQLi, XXE, deserialization, SSRF, SSTI, command injection, file upload)
- ■ Professional penetration testing report writing skills
- ■ Active OWASP LA and Null Space Labs participation (26+ hours total)

Quarter 2: Modern APIs + DevSecOps

(Weeks 42-54)

September 15 - December 14, 2026 | 234 Hours Total (18h/week)

Primary Objectives:

- Master GraphQL security exploitation and defense
- Achieve gRPC and microservices security expertise
- Build comprehensive API security testing framework
- Develop CI/CD security automation pipeline
- Integrate DevSecOps practices into all tools

Week 42: GraphQL Security Fundamentals (18 hours)

Books:

- GraphQL in Action by Samer Buna (Manning, 2021), Chapters 1-4: GraphQL Basics (pp. 1-120)
- Hacking APIs by Corey J. Ball (No Starch Press, 2022), Chapter 8: GraphQL (pp. 191-218)

Online Resources:

- OWASP GraphQL Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/GraphQLCheatSheet.html
- GraphQL Security Best Practices
- https://graphql.org/learn/best-practices/
- PortSwigger GraphQL Labs
- https://portswigger.net/web-security/graphql

Practice (10 hours):

- Build GraphQL introspection analyzer
- Create GraphQL injection detector
- Implement GraphQL rate limiting bypass toolkit
- Research GraphQL batching attacks

Labs (4 hours):

- PortSwigger GraphQL labs (complete all)
- DVGA (Damn Vulnerable GraphQL Application)

Writing (2 hours):

- Blog Post: "GraphQL Security: Common Vulnerabilities and Exploitation Techniques"

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- GraphQL security testing toolkit
- Blog post published

Week 43: GraphQL Advanced Exploitation (18 hours)

Books:

- GraphQL in Action by Samer Buna (Manning, 2021), Chapters 5-8: Advanced Features (pp. 121-240)
- Microservices Security in Action by Prabath Siriwardena (Manning, 2020), Chapter 2: First Steps (pp. 19-46)

Online Resources:

- GraphQL Vulnerability Research
- https://blog.yeswehack.com/yeswerhackers/how-exploit-graphql-endpoint-bug-bounty/
- OWASP API Security Top 10
- https://owasp.org/API-Security/
- GraphQL Authentication & Authorization
- https://www.apollographql.com/docs/apollo-server/security/authentication/

Practice (10 hours):

- Build GraphQL authorization bypass detector
- Create GraphQL depth limit tester
- Implement GraphQL batching attack toolkit
- Research GraphQL subscription vulnerabilities

Labs (4 hours):

- DVGA advanced scenarios
- Real-world GraphQL API testing (bug bounty style)

Writing (2 hours):

- Blog Post: "Advanced GraphQL Attacks: Authorization Bypass & Query Complexity"

Community (2 hours):

- OWASP LA meeting

Deliverables:

• Advanced GraphQL exploitation toolkit

• Blog post published

Week 44: gRPC Security & Protocol Buffers (18 hours)

Books:

• Microservices Security in Action by Prabath Siriwardena (Manning, 2020), Chapter 3: Securing microservices (pp. 47-78)

• API Security in Action by Neil Madden (Manning, 2020), Chapter 11: Microservice APIs (pp. 357-392) [Review]

Online Resources:

• gRPC Security Documentation

• https://grpc.io/docs/guides/auth/

• Protocol Buffers Security Considerations

• https://developers.google.com/protocol-buffers/docs/security

• OWASP gRPC Security Guide

• https://owasp.org/www-project-devsecops-guideline/latest/02c-API-Security

Practice (10 hours):

• Build gRPC security scanner

• Create protocol buffer analyzer

• Implement gRPC authentication testing toolkit

• Research gRPC reflection exploitation

Labs (4 hours):

• gRPC security CTF challenges

• Microservices security scenarios

Writing (2 hours):

• Blog Post: "gRPC Security: Testing and Exploitation Techniques"

Community (2 hours):

• Null Space Labs engagement

Deliverables:

• gRPC security testing toolkit

• Blog post published

Week 45: Service Mesh Security (Istio/Linkerd) (18 hours)

Books:

• Microservices Security in Action by Prabath Siriwardena (Manning, 2020), Chapter 9: Securing microservices communication (pp. 247-286)

• Container Security by Liz Rice (O'Reilly, 2020), Chapter 9: Service Mesh (pp. 217-240)

Online Resources:

• Istio Security Documentation

• https://istio.io/latest/docs/concepts/security/

• Linkerd Security Best Practices

• https://linkerd.io/2/features/security/

• OWASP Service Mesh Security

• https://github.com/OWASP/www-project-kubernetes-top-ten

Practice (10 hours):

• Set up local Istio environment

• Build service mesh configuration auditor

• Create mTLS verification toolkit

• Research service mesh policy bypass techniques

Labs (4 hours):

• Istio security labs

• Service mesh misconfigurations CTF

Writing (2 hours):

• Blog Post: "Service Mesh Security: Istio mTLS and Policy Enforcement"

Community (2 hours):

• OWASP LA engagement

Deliverables:

• Service mesh security auditor

• Blog post published

Week 46: REST API Security Deep Dive (18 hours)

Books:

• Hacking APIs by Corey J. Ball (No Starch Press, 2022), Chapters 4-7: API Discovery to Auth Attacks (pp. 73-190) [Review & Advanced]

• API Security in Action by Neil Madden (Manning, 2020), Selected chapters review

Online Resources:

• OWASP API Security Project

• https://owasp.org/www-project-api-security/

• REST API Security Best Practices

• https://cheatsheetseries.owasp.org/cheatsheets/RESTSecurityCheatSheet.html

- API Pentesting Methodology
- https://university.apisec.ai/

Practice (10 hours):

- Build comprehensive REST API security scanner
- Create API discovery automation toolkit
- Implement API fuzzing framework
- Research modern API authentication bypass techniques

Labs (4 hours):

- Hack-the-Box API challenges
- Real-world API testing practice

Writing (2 hours):

- Blog Post: "REST API Security Testing: A Comprehensive Methodology"

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Enterprise-grade REST API scanner
- Blog post published

Week 47: CI/CD Pipeline Security (18 hours)

Books:

- Securing DevOps by Julien Vehent (Manning, 2018), Chapters 3-5: Continuous Security (pp. 39-130)
- The DevSecOps Playbook by Sean Mack (Wiley, 2024), Chapters 2-4: Pipeline Security (pp. 25-98)

Online Resources:

- OWASP DevSecOps Guideline
- https://owasp.org/www-project-devsecops-guideline/
- GitHub Actions Security Best Practices
- https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions
- GitLab CI/CD Security
- https://docs.gitlab.com/ee/ci/security/

Practice (10 hours):

- Build CI/CD security gate framework
- Create pipeline configuration scanner
- Implement secrets detection in CI/CD
- Research supply chain attack vectors in pipelines

Labs (4 hours):

- CI/CD security CTF challenges
- Pipeline exploitation scenarios

Writing (2 hours):

- Blog Post: "Securing CI/CD Pipelines: From Code to Production"

Community (2 hours):

- OWASP LA meeting

Deliverables:

- CI/CD security automation framework
- Blog post published

Week 48: Container Security Deep Dive (18 hours)

Books:

- Container Security by Liz Rice (O'Reilly, 2020), Chapters 5-6: Container Runtime & Host Security (pp. 115-164)
- Kubernetes Security by Liz Rice & Michael Hausenblas (O'Reilly, 2021), Chapter 6: Image Security (pp. 157-184)

Online Resources:

- OWASP Container Security Verification Standard
- https://github.com/OWASP/Container-Security-Verification-Standard
- Docker Security Best Practices
- https://docs.docker.com/develop/security-best-practices/
- Container Escape Techniques
- https://book.hacktricks.xyz/linux-hardening/privilege-escalation/docker-security

Practice (10 hours):

- Build container security scanner (runtime analysis)
- Create container escape detection toolkit
- Implement Docker socket security auditor
- Research container breakout techniques

Labs (4 hours):

- Container escape CTF challenges
- Docker security misconfigurations

Writing (2 hours):

- Blog Post: "Container Security: Runtime Protection and Escape Prevention"

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Container runtime security scanner
- Blog post published

Week 49: Supply Chain Security (18 hours)

Books:

- Securing DevOps by Julien Vehent (Manning, 2018), Chapter 6: Securing Software Supply Chain (pp. 131-166)
- The DevSecOps Playbook by Sean Mack (Wiley, 2024), Chapter 7: Supply Chain (pp. 155-188)

Online Resources:

- OWASP Software Component Analysis
- https://owasp.org/www-community/ComponentAnalysis
- SLSA Framework
- https://slsa.dev/
- SBOM (Software Bill of Materials)
- https://www.cisa.gov/sbom

Practice (10 hours):

- Build dependency vulnerability scanner
- Create SBOM generator and analyzer
- Implement supply chain attack detector
- Research package repository attacks (PyPI, npm)

Labs (4 hours):

- Supply chain security CTF
- Dependency confusion attack scenarios

Writing (2 hours):

- Blog Post: "Software Supply Chain Security: Detection and Prevention"

Community (2 hours):

- OWASP LA engagement

Deliverables:

- Supply chain security toolkit
- Blog post published

Week 50: Secrets Management & Vault (18 hours)

Books:

- Securing DevOps by Julien Vehent (Manning, 2018), Chapter 7: Securing Communications (pp. 167-206)

• Microservices Security in Action by Prabath Siriwardena (Manning, 2020), Chapter 11: Secrets management (pp. 335-366)

Online Resources:

• HashiCorp Vault Documentation

• https://developer.hashicorp.com/vault/docs

• OWASP Secrets Management Cheat Sheet

• https://cheatsheetseries.owasp.org/cheatsheets/SecretsManagementCheatSheet.html

• AWS Secrets Manager Best Practices

• https://docs.aws.amazon.com/secretsmanager/latest/userguide/best-practices.html

Practice (10 hours):

• Build secrets scanner (hardcoded credentials detector)

• Create Vault integration framework

• Implement secrets rotation auditor

• Research secrets sprawl in microservices

Labs (4 hours):

• Vault security configuration

• Secrets management CTF challenges

Writing (2 hours):

• Blog Post: "Secrets Management in Cloud-Native Applications: Best Practices"

Community (2 hours):

• Null Space Labs engagement

Deliverables:

• Enterprise secrets management toolkit

• Blog post published

Week 51: DevSecOps Dashboard Development (18 hours)

Books:

• The DevSecOps Playbook by Sean Mack (Wiley, 2024), Chapter 8: Metrics & Monitoring (pp. 189-224)

• Securing DevOps by Julien Vehent (Manning, 2018), Chapter 8: Continuous Monitoring (pp. 207-238)

Online Resources:

• OWASP Security Metrics

• https://owasp.org/www-community/OWASPMetricsProject

• ELK Stack Security

• https://www.elastic.co/guide/en/elasticsearch/reference/current/security-settings.html

• Prometheus & Grafana for Security

• https://grafana.com/grafana/dashboards/

Practice (10 hours):

- Build comprehensive DevSecOps dashboard
- Create security metrics aggregator
- Implement vulnerability trend analysis
- Research security observability patterns

Labs (4 hours):

- Dashboard development & testing
- Metrics collection integration

Writing (2 hours):

- Blog Post: "Building a DevSecOps Security Dashboard: Metrics That Matter"

Community (2 hours):

- OWASP LA meeting

Deliverables:

- Production-ready DevSecOps dashboard
- Blog post published

Week 52: API Fuzzing & Chaos Engineering (18 hours)

Books:

- Hacking APIs by Corey J. Ball (No Starch Press, 2022), Chapters 11-13: Fuzzing & Exploitation (pp. 259-322)
- Microservices Security in Action by Prabath Siriwardena (Manning, 2020), Chapter 12: Testing (pp. 367-400)

Online Resources:

- OWASP API Security Testing
- https://owasp.org/www-project-web-security-testing-guide/latest/4-WebApplicationSecurityTesting/12-API Testing/
- API Fuzzing Tools Comparison
- https://github.com/topics/api-fuzzing
- Chaos Engineering Security
- https://principlesofchaos.org/

Practice (10 hours):

- Build custom API fuzzer
- Create GraphQL/gRPC fuzzing framework
- Implement chaos engineering security tests
- Research automated vulnerability discovery

Labs (4 hours):

- API fuzzing against test applications
- Chaos engineering experiments

Writing (2 hours):

- Blog Post: "API Fuzzing: Automated Vulnerability Discovery in Modern APIs"

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Advanced API fuzzing toolkit
- Blog post published

Week 53: API Gateway Security (18 hours)

Books:

- Microservices Security in Action by Prabath Siriwardena (Manning, 2020), Chapter 7: Securing APIs with Kong (pp. 183-220)
- API Security in Action by Neil Madden (Manning, 2020), Chapter 12: API Gateways (pp. 393-426)

Online Resources:

- API Gateway Security Best Practices
- https://konghq.com/learning-center/api-gateway/api-gateway-security
- AWS API Gateway Security
- https://docs.aws.amazon.com/apigateway/latest/developerguide/security.html
- OWASP API Gateway Security
- https://cheatsheetseries.owasp.org/cheatsheets/APISecurityCheatSheet.html

Practice (10 hours):

- Build API gateway security auditor
- Create rate limiting bypass detector
- Implement API gateway policy analyzer
- Research API gateway misconfigurations

Labs (4 hours):

- Kong/NGINX API Gateway security labs
- AWS API Gateway exploitation

Writing (2 hours):

- Blog Post: "API Gateway Security: Common Misconfigurations and Exploitation"

Community (2 hours):

- OWASP LA engagement

Deliverables:

- API gateway security toolkit
- Blog post published

Week 54: Q2 Review & Portfolio Update (18 hours)

Activities:

- Review all Q2 learnings and tools
- Update portfolio website with Q2 deliverables
- Prepare comprehensive API security testing methodology document
- Create DevSecOps best practices guide
- Document all tools and frameworks built in Q2

Practice (12 hours):

- Polish all Q2 tools for production use
- Write comprehensive documentation
- Create demo videos for key tools
- Update GitHub repositories

Writing (4 hours):

- Comprehensive Q2 summary blog post
- "Year in Review" mid-year update

Community (2 hours):

- Q2 final OWASP LA meeting

Deliverables:

- Complete Q2 portfolio update
- All Q2 tools production-ready with documentation
- Mid-year review blog post

Quarter 2 Summary: Success Metrics

Completed by December 14, 2026:

- ■ GraphQL security expertise (introspection, injection, batching, authorization bypass)
- ■ gRPC and Protocol Buffers security mastery
- ■ Service mesh security (Istio/Linkerd) expertise
- ■ Comprehensive REST API security testing framework
- ■ CI/CD pipeline security automation
- ■ Container security deep knowledge (runtime, escapes)
- ■ Supply chain security toolkit
- ■ Secrets management framework (Vault integration)
- ■ Production DevSecOps dashboard

- ■ Advanced API fuzzing capabilities
- ■ API gateway security expertise
- ■ 13 technical blog posts published (total: 26 Year-to-Date)
- ■ 13+ production security tools built (total: 26+ Year-to-Date)
- ■ Active community participation (26+ hours)

Quarter 3: Custom Tooling + SAST/DAST

(Weeks 55-67)

December 15, 2026 - March 15, 2027 | 234 Hours Total (18h/week)

Primary Objectives:

- Build custom SAST engine with pattern recognition
- Develop advanced DAST framework with AI-powered testing
- Master static code analysis for multiple languages
- Achieve AST (Abstract Syntax Tree) manipulation expertise
- Create production-grade security automation platform

Week 55: SAST Fundamentals & AST Parsing (18 hours)

Books:

- Engineering a Compiler by Cooper & Torczon (Morgan Kaufmann, 2011), Chapters 1-2: Compiler Overview (pp. 1-78)
- The Art of Software Security Assessment by Dowd, McDonald & Schuh (Addison-Wesley, 2006), Chapter 7: Program Building Blocks (pp. 289-344)

Online Resources:

- Semgrep Documentation
- https://semgrep.dev/docs/
- OWASP Code Review Guide
- https://owasp.org/www-project-code-review-guide/
- Tree-sitter Parser Documentation
- https://tree-sitter.github.io/tree-sitter/

Practice (10 hours):

- Build Python AST analyzer
- Create basic pattern-matching SAST engine
- Implement syntax tree traversal algorithms
- Research compiler design principles

Labs (4 hours):

- Semgrep rule writing exercises

- AST manipulation practice

Writing (2 hours):

- Blog Post: "Building a Static Code Analyzer: Understanding Abstract Syntax Trees"

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Basic SAST engine prototype
- Blog post published

Week 56: Advanced SAST Pattern Recognition (18 hours)

Books:

- Engineering a Compiler by Cooper & Torczon (Morgan Kaufmann, 2011), Chapter 5: Syntax-Directed Translation (pp. 247-306)
- The Art of Software Security Assessment by Dowd, McDonald & Schuh (Addison-Wesley, 2006), Chapter 8: Strings & Metacharacters (pp. 345-412)

Online Resources:

- Semgrep Advanced Patterns
- https://semgrep.dev/docs/writing-rules/pattern-syntax/
- CodeQL Documentation
- https://codeql.github.com/docs/
- OWASP SAMM (Software Assurance Maturity Model)
- https://owaspsamm.org/

Practice (10 hours):

- Build dataflow analysis engine
- Create taint tracking implementation
- Implement inter-procedural analysis
- Research advanced pattern matching techniques

Labs (4 hours):

- CodeQL query development
- Complex vulnerability pattern detection

Writing (2 hours):

- Blog Post: "Advanced SAST: Dataflow Analysis and Taint Tracking"

Community (2 hours):

- OWASP LA meeting

Deliverables:

• Advanced SAST engine with dataflow analysis
• Blog post published

Week 57: Multi-Language SAST Support (18 hours)

Books:

• Crafting Interpreters by Robert Nystrom (Free online), Chapters 1-8: Scanning & Parsing (pp. 1-150)
• https://craftinginterpreters.com/
• The Art of Software Security Assessment by Dowd, McDonald & Schuh (Addison-Wesley, 2006), Chapter 6: C Language Issues (pp. 209-288)

Online Resources:

• Language-Specific Security Patterns
• Multiple OWASP Cheat Sheets per language
• Tree-sitter Language Parsers
• https://github.com/tree-sitter
• Polyglot Analysis Techniques
• Academic papers on multi-language analysis

Practice (10 hours):

• Add JavaScript/TypeScript support to SAST engine
• Implement Java analysis capabilities
• Create Go security pattern library
• Research language-agnostic vulnerability patterns

Labs (4 hours):

• Multi-language vulnerability detection
• Cross-language taint flow analysis

Writing (2 hours):

• Blog Post: "Building a Multi-Language SAST Engine: Architecture and Challenges"

Community (2 hours):

• Null Space Labs engagement

Deliverables:

• Multi-language SAST engine
• Blog post published

Week 58: DAST Fundamentals & Crawling (18 hours)

Books:

- The Tangled Web by Michal Zalewski (No Starch Press, 2011), Chapters 1-5: Browser Mechanics (pp. 1-120)
- Hacking APIs by Corey J. Ball (No Starch Press, 2022), Chapter 3: Discovery (pp. 47-72) [Review]

Online Resources:

- OWASP ZAP Documentation
- https://www.zaproxy.org/docs/
- Burp Suite API Documentation
- https://portswigger.net/burp/documentation/desktop/tools
- Web Crawling Best Practices
- https://developers.google.com/search/docs/crawling-indexing

Practice (10 hours):

- Build intelligent web crawler
- Create AJAX/SPA crawling capabilities
- Implement authentication-aware crawling
- Research modern web application architecture discovery

Labs (4 hours):

- OWASP ZAP automation
- SPA crawling challenges

Writing (2 hours):

- Blog Post: "Building a Modern Web Crawler for Security Testing"

Community (2 hours):

- OWASP LA engagement

Deliverables:

- Production-grade web crawler
- Blog post published

Week 59: DAST Vulnerability Detection (18 hours)

Books:

- The Tangled Web by Michal Zalewski (No Starch Press, 2011), Chapters 6-10: Security Concepts (pp. 121-280)
- Real-World Bug Hunting by Peter Yaworski (No Starch Press, 2019), Chapters 3-6: Common Vulnerabilities (pp. 51-150)

Online Resources:

- OWASP Testing Guide
- https://owasp.org/www-project-web-security-testing-guide/
- Nuclei Templates

- https://github.com/projectdiscovery/nuclei-templates
- Active Vulnerability Scanning Techniques
- Multiple research papers

Practice (10 hours):

- Build DAST vulnerability scanner core
- Create SQLi, XSS, CSRF detection modules
- Implement false positive reduction algorithms
- Research intelligent fuzzing techniques

Labs (4 hours):

- DAST scanner testing against vulnerable apps
- False positive analysis

Writing (2 hours):

- Blog Post: "Building a DAST Scanner: Architecture and Vulnerability Detection"

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Core DAST scanner with multiple detection modules
- Blog post published

Week 60: AI-Powered Security Testing (18 hours)

Books:

- Deep Learning for Coders with fastai and PyTorch by Howard & Gugger (O'Reilly, 2020), Chapters 1-4: Basics (pp. 1-140)
- Real-World Bug Hunting by Peter Yaworski (No Starch Press, 2019), Chapters 10-13: Advanced Techniques (pp. 249-354)

Online Resources:

- ML for Security Testing
- https://arxiv.org/list/cs.CR/recent (Security & Cryptography papers)
- TensorFlow Security
- https://www.tensorflow.org/responsibleai/fairnessindicators/guide
- AI-Powered Fuzzing Techniques
- Academic research papers

Practice (10 hours):

- Build ML-based payload generator
- Create intelligent test case prioritization
- Implement anomaly detection for vulnerabilities

- Research LLM-assisted security testing

Labs (4 hours):

- Train vulnerability prediction models
- Test AI-powered fuzzing

Writing (2 hours):

- Blog Post: "AI-Powered Security Testing: Machine Learning for Vulnerability Discovery"

Community (2 hours):

- OWASP LA meeting

Deliverables:

- AI-enhanced security testing framework
- Blog post published

Week 61: API Security Testing Automation (18 hours)

Books:

- Hacking APIs by Corey J. Ball (No Starch Press, 2022), Complete review of key chapters
- API Security in Action by Neil Madden (Manning, 2020), Complete review of key chapters

Online Resources:

- OWASP API Security Testing Guide
- https://owasp.org/www-project-api-security/
- Postman Security Testing
- https://learning.postman.com/docs/writing-scripts/test-scripts/
- OpenAPI Specification Security
- https://spec.openapis.org/oas/latest.html

Practice (10 hours):

- Build comprehensive API security testing framework
- Create OpenAPI specification parser & tester
- Implement automated API authentication testing
- Research API versioning security issues

Labs (4 hours):

- Automated API security testing scenarios
- OpenAPI specification security analysis

Writing (2 hours):

- Blog Post: "Automating API Security Testing: From OpenAPI to Comprehensive Coverage"

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Enterprise API security testing automation
- Blog post published

Week 62: Security Reporting & Visualization (18 hours)

Books:

- The Art of Software Security Assessment by Dowd, McDonald & Schuh (Addison-Wesley, 2006), Chapter 3: Operational Review (pp. 93-136)
- Threat Modeling: Designing for Security by Adam Shostack (Wiley, 2014), Chapter 11: Reporting & Tracking (pp. 233-256)

Online Resources:

- Security Report Templates
- https://github.com/juliocesarfort/public-pentesting-reports
- CVSS Documentation
- https://www.first.org/cvss/
- Security Metrics & KPIs
- OWASP metrics documentation

Practice (10 hours):

- Build automated report generation system
- Create vulnerability visualization dashboard
- Implement CVSS score calculator
- Research effective security communication

Labs (4 hours):

- Report generation testing
- Dashboard development

Writing (2 hours):

- Blog Post: "Security Reporting Done Right: From Findings to Actionable Insights"

Community (2 hours):

- OWASP LA engagement

Deliverables:

- Automated security reporting platform
- Blog post published

Week 63: Infrastructure Security Scanning (18 hours)

Books:

• Cloud Security and Privacy by Mather, Kumaraswamy & Latif (O'Reilly, 2009), Chapters 1-4: Cloud Fundamentals (pp. 1-110)

• Kubernetes Security by Liz Rice & Michael Hausenblas (O'Reilly, 2021), Review key chapters

Online Resources:

• Cloud Security Posture Management (CSPM)

• https://csrc.nist.gov/publications/detail/sp/800-204d/final

• Infrastructure Security Scanning Tools

• Prowler, ScoutSuite, CloudSploit documentation

• CIS Benchmarks

• https://www.cisecurity.org/cis-benchmarks/

Practice (10 hours):

• Build infrastructure security scanner

• Create CIS benchmark compliance checker

• Implement cloud misconfigurations detector

• Research infrastructure as code security

Labs (4 hours):

• Infrastructure scanning scenarios

• CIS compliance testing

Writing (2 hours):

• Blog Post: "Infrastructure Security Scanning: Detecting Cloud Misconfigurations"

Community (2 hours):

• Null Space Labs engagement

Deliverables:

• Infrastructure security scanner

• Blog post published

Week 64: Security Orchestration Platform (18 hours)

Books:

• Security Engineering by Ross Anderson (Wiley, 3rd Edition, 2020), Chapter 27: Security Operations (pp. 745-778) [Free online]

• https://www.cl.cam.ac.uk/~rja14/book.html

• Building Secure & Reliable Systems by Adkins, Beyer, Blankinship, et al (O'Reilly, 2020), Chapter 8: Security Architecture (pp. 135-158) [Free online]

• https://static.googleusercontent.com/media/sre.google/en//static/pdf/buildingsecureandreliablesystems.pdf

Online Resources:

• SOAR Platforms Overview

- https://www.gartner.com/en/information-technology/glossary/security-orchestration-automation-response-soar
- Security Automation Best Practices
- https://owasp.org/www-project-devsecops-guideline/
- Integration Patterns
- Multiple vendor documentation

Practice (10 hours):

- Build security orchestration platform
- Create tool integration framework
- Implement workflow automation
- Research SOAR architecture patterns

Labs (4 hours):

- Platform integration testing
- Workflow automation scenarios

Writing (2 hours):

- Blog Post: "Building a Security Orchestration Platform: Integrating Multiple Tools"

Community (2 hours):

- OWASP LA meeting

Deliverables:

- Security orchestration platform
- Blog post published

Week 65: Custom Security Policies & Rules (18 hours)

Books:

- Secure by Design by Johnsson, Deogun & Sawano (Manning, 2019), Complete review
- Security Engineering by Ross Anderson (Wiley, 3rd Edition, 2020), Chapter 10: Access Control (pp. 253-290) [Free online]

Online Resources:

- Policy as Code
- https://www.openpolicyagent.org/docs/latest/
- Custom Rule Development
- Semgrep, CodeQL, Checkov documentation
- Security Policy Frameworks
- Multiple standards (NIST, ISO)

Practice (10 hours):

- Build policy engine for custom rules

• Create language-specific security patterns
• Implement policy violation detector
• Research policy as code best practices

Labs (4 hours):

• Custom policy testing
• Rule effectiveness analysis

Writing (2 hours):

• Blog Post: "Implementing Policy as Code: Custom Security Rules at Scale"

Community (2 hours):

• Null Space Labs engagement

Deliverables:

• Custom policy engine
• Blog post published

Week 66: CVE Research & Discovery (18 hours)

Books:

• Real-World Bug Hunting by Peter Yaworski (No Starch Press, 2019), Complete review
• The Art of Software Security Assessment by Dowd, McDonald & Schuh (Addison-Wesley, 2006), Chapter 4: Application Review Process (pp. 137-176)

Online Resources:

• CVE Program Documentation
• https://www.cve.org/ResourcesSupport/AllResources
• Vulnerability Disclosure Policies
• https://cheatsheetseries.owasp.org/cheatsheets/VulnerabilityDisclosureCheatSheet.html
• Bug Bounty Platforms
• HackerOne, Bugcrowd documentation

Practice (10 hours):

• Research potential CVEs in open source projects
• Build vulnerability discovery methodology
• Create exploit PoC development framework
• Research responsible disclosure practices

Labs (4 hours):

• CVE research practice
• Exploit development scenarios

Writing (2 hours):

- Blog Post: "CVE Research: From Discovery to Responsible Disclosure"

Community (2 hours):

- OWASP LA engagement

Deliverables:

- CVE research methodology
- 1-2 potential CVE submissions
- Blog post published

Week 67: Q3 Review & Tool Integration (18 hours)

Activities:

- Review all Q3 learnings and tools
- Update portfolio website with Q3 deliverables
- Integrate all SAST/DAST tools into unified platform
- Create comprehensive security automation documentation
- Polish all Q3 tools for production use

Practice (12 hours):

- Final integration of all Q3 tools
- Write comprehensive documentation
- Create demo videos and tutorials
- Update GitHub repositories

Writing (4 hours):

- Comprehensive Q3 summary blog post
- Technical whitepaper on custom security automation

Community (2 hours):

- Q3 final OWASP LA meeting

Deliverables:

- Complete Q3 portfolio update
- All Q3 tools production-ready
- Q3 review blog post
- Security automation whitepaper

Quarter 3 Summary: Success Metrics

Completed by March 15, 2027:

- ■ Custom SAST engine with multi-language support

- ■ Advanced DAST framework with intelligent crawling
- ■ AI-powered security testing capabilities
- ■ Comprehensive API security testing automation
- ■ Security reporting & visualization platform
- ■ Infrastructure security scanner
- ■ Security orchestration platform (SOAR)
- ■ Custom policy engine
- ■ CVE research expertise (1-2 submissions)
- ■ 13 technical blog posts published (total: 39 Year-to-Date)
- ■ 13+ production security tools built (total: 39+ Year-to-Date)
- ■ Active community participation (26+ hours)

Quarter 4: Enterprise Architecture + Thought Leadership

(Weeks 68-80)

March 16 - June 14, 2027 | 182 Hours Total (14h/week)

Primary Objectives:

- Master enterprise security architecture principles
- Develop thought leadership presence (conference talks, advanced blog posts)
- Build security architecture review framework
- Achieve advanced threat modeling expertise
- Prepare for principal engineer / security architect roles

Week 68: Security Architecture Fundamentals (14 hours)

Books:

- Security Engineering by Ross Anderson (Wiley, 3rd Edition, 2020), Chapters 1-3: Security Introduction (pp. 1-80) [Free online]
- Threat Modeling: Designing for Security by Adam Shostack (Wiley, 2014), Chapters 1-4: Fundamentals (pp. 1-100)

Online Resources:

- NIST Cybersecurity Framework
- https://www.nist.gov/cyberframework
- OWASP Application Security Verification Standard (ASVS)
- https://owasp.org/www-project-application-security-verification-standard/
- Enterprise Security Architecture Patterns
- Academic papers and industry whitepapers

Practice (8 hours):

- Build security architecture review framework
- Create architecture decision record (ADR) templates
- Implement security architecture documentation system
- Research enterprise security patterns

Writing (4 hours):

- Blog Post: "Enterprise Security Architecture: Principles and Patterns"
- Start conference talk proposal on AppSec topics

Community (2 hours):

- OWASP LA engagement

Deliverables:

- Security architecture review framework
- Blog post published
- Conference talk proposal draft

Week 69: Advanced Threat Modeling (14 hours)

Books:

- Threat Modeling: Designing for Security by Adam Shostack (Wiley, 2014), Chapters 5-10: Advanced Topics (pp. 101-232)
- Building Secure & Reliable Systems by Adkins, Beyer, Blankinship, et al (O'Reilly, 2020), Chapter 14: Threat Modeling (pp. 245-268) [Free online]

Online Resources:

- OWASP Threat Modeling Toolkit
- https://owasp.org/www-community/ThreatModeling
- Microsoft Threat Modeling Tool
- https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool
- STRIDE vs PASTA vs LINDDUN Comparison
- Academic research papers

Practice (8 hours):

- Build advanced threat modeling framework
- Create automated threat model generation
- Implement STRIDE on steroids methodology
- Research ML-assisted threat modeling

Writing (4 hours):

- Blog Post: "Advanced Threat Modeling: Beyond Basic STRIDE"
- Continue conference talk development

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Advanced threat modeling framework
- Blog post published

Week 70: Zero Trust Architecture (14 hours)

Books:

- Zero Trust Networks by Gilman & Barth (O'Reilly, 2017), Chapters 1-5: Zero Trust Concepts (pp. 1-120)
- Building Secure & Reliable Systems by Adkins, Beyer, Blankinship, et al (O'Reilly, 2020), Chapter 6: Design Tradeoffs (pp. 105-134) [Free online]

Online Resources:

- NIST Zero Trust Architecture
- https://csrc.nist.gov/publications/detail/sp/800-207/final
- Zero Trust Maturity Model
- https://www.cisa.gov/zero-trust-maturity-model
- Zero Trust Implementation Guides
- Multiple vendor resources (Google BeyondCorp, etc.)

Practice (8 hours):

- Build zero trust architecture assessment framework
- Create zero trust maturity model scorer
- Implement microsegmentation analyzer
- Research zero trust implementation patterns

Writing (4 hours):

- Blog Post: "Implementing Zero Trust Architecture: Practical Patterns"
- Refine conference talk proposal

Community (2 hours):

- OWASP LA engagement

Deliverables:

- Zero trust assessment framework
- Blog post published

Week 71: Security Architecture Review Process (14 hours)

Books:

- Security Engineering by Ross Anderson (Wiley, 3rd Edition, 2020), Chapters 28-29: Assurance & Evaluation (pp. 779-832) [Free online]

- The Art of Software Security Assessment by Dowd, McDonald & Schuh (Addison-Wesley, 2006), Chapters 1-2: Assessment Process (pp. 1-92)

Online Resources:

- Architecture Review Board Best Practices
- Industry whitepapers
- OWASP Architecture Review Cheat Sheet
- https://cheatsheetseries.owasp.org/cheatsheets/
- Security Design Reviews
- Multiple vendor methodologies

Practice (8 hours):

- Build architecture review checklist generator
- Create security design pattern catalog
- Implement automated architecture analysis
- Research architecture anti-patterns

Writing (4 hours):

- Blog Post: "Conducting Effective Security Architecture Reviews"
- Finalize conference talk abstract

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Architecture review methodology
- Blog post published
- Conference talk submitted

Week 72: Secure-by-Design Principles (14 hours)

Books:

- Secure by Design by Johnsson, Deogun & Sawano (Manning, 2019), Complete deep review
- Building Secure & Reliable Systems by Adkins, Beyer, Blankinship, et al (O'Reilly, 2020), Chapter 7: Design Patterns (pp. 135-158) [Free online]

Online Resources:

- OWASP Secure Coding Practices
- https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/
- Security Design Principles
- https://www.ncsc.gov.uk/collection/developers-collection/principles
- Defense in Depth Strategies
- NIST and SANS resources

Practice (8 hours):

- Build secure design pattern library
- Create secure-by-design assessment tool
- Implement design flaw detector
- Research immutability and domain-driven design for security

Writing (4 hours):

- Blog Post: "Secure-by-Design: Implementing Security from the Ground Up"
- Prepare conference talk slides

Community (2 hours):

- OWASP LA meeting

Deliverables:

- Secure design pattern library
- Blog post published

Week 73: Microservices Security Architecture (14 hours)

Books:

- Microservices Security in Action by Prabath Siriwardena (Manning, 2020), Complete deep review
- Building Secure & Reliable Systems by Adkins, Beyer, Blankinship, et al (O'Reilly, 2020), Chapter 11: Distributed Systems (pp. 193-218) [Free online]

Online Resources:

- Microservices Security Best Practices
- https://owasp.org/www-project-devsecops-guideline/latest/02d-Microservices-Security
- Service Mesh Architecture
- Istio, Linkerd architecture documentation
- Distributed Systems Security
- Academic papers

Practice (8 hours):

- Build microservices security architecture analyzer
- Create service-to-service auth validator
- Implement distributed tracing security analyzer
- Research sidecar pattern security implications

Writing (4 hours):

- Blog Post: "Microservices Security Architecture: Patterns and Anti-Patterns"
- Practice conference talk delivery

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Microservices security analyzer
- Blog post published

Week 74: Cloud-Native Security Architecture (14 hours)

Books:

- Cloud Security and Privacy by Mather, Kumaraswamy & Latif (O'Reilly, 2009), Chapters 5-8: Security Architecture (pp. 111-240)
- Kubernetes Security by Liz Rice & Michael Hausenblas (O'Reilly, 2021), Complete deep review

Online Resources:

- CNCF Security Whitepaper
- https://www.cncf.io/blog/2020/11/18/announcing-the-cloud-native-security-whitepaper/
- Cloud Security Alliance Resources
- https://cloudsecurityalliance.org/
- Multi-Cloud Security Architecture
- Industry research papers

Practice (8 hours):

- Build cloud-native security architecture framework
- Create multi-cloud security analyzer
- Implement cloud security posture validator
- Research serverless security architecture

Writing (4 hours):

- Blog Post: "Cloud-Native Security Architecture: Kubernetes and Beyond"
- Refine conference talk based on practice feedback

Community (2 hours):

- OWASP LA engagement

Deliverables:

- Cloud-native security framework
- Blog post published

Week 75: Supply Chain Security Architecture (14 hours)

Books:

- Building Secure & Reliable Systems by Adkins, Beyer, Blankinship, et al (O'Reilly, 2020), Chapter 12: Supply Chain Security (pp. 219-244) [Free online]
- Securing DevOps by Julien Vehent (Manning, 2018), Complete deep review

Online Resources:

- SLSA Framework
- https://slsa.dev/spec/v1.0/
- NIST Secure Software Development Framework
- https://csrc.nist.gov/Projects/ssdf
- Software Bill of Materials (SBOM)
- https://www.cisa.gov/sbom

Practice (8 hours):

- Build supply chain security architecture framework
- Create SLSA compliance checker
- Implement build provenance validator
- Research reproducible builds security

Writing (4 hours):

- Blog Post: "Supply Chain Security Architecture: SLSA and Beyond"
- Final conference talk rehearsal

Community (2 hours):

- Null Space Labs engagement

Deliverables:

- Supply chain security framework
- Blog post published

Week 76: Security Metrics & KPIs (14 hours)

Books:

- Security Engineering by Ross Anderson (Wiley, 3rd Edition, 2020), Chapter 26: Security Economics (pp. 711-744) [Free online]
- The DevSecOps Playbook by Sean Mack (Wiley, 2024), Complete deep review

Online Resources:

- OWASP Security Metrics
- https://owasp.org/www-community/OWASPMetricsProject
- BSIMM (Building Security In Maturity Model)
- https://www.bsimm.com/
- Security Program Metrics
- Industry research and whitepapers

Practice (8 hours):

- Build security metrics dashboard
- Create KPI tracking framework

• Implement security ROI calculator

• Research security program maturity models

Writing (4 hours):

• Blog Post: "Security Metrics That Matter: Measuring AppSec Program Success"

• Prepare conference talk Q&A;

Community (2 hours):

• OWASP LA meeting

Deliverables:

• Security metrics framework

• Blog post published

Week 77: Incident Response Architecture (14 hours)

Books:

• Security Engineering by Ross Anderson (Wiley, 3rd Edition, 2020), Chapter 27: Security Operations (pp. 745-778) [Free online]

• The Art of Software Security Assessment by Dowd, McDonald & Schuh (Addison-Wesley, 2006), Chapter 2: Design Review (pp. 49-92)

Online Resources:

• NIST Incident Response Guide

• https://csrc.nist.gov/publications/detail/sp/800-61/rev-2/final

• SANS Incident Response Process

• https://www.sans.org/white-papers/

• Security Incident Response Automation

• SOAR platform documentation

Practice (8 hours):

• Build incident response playbook generator

• Create automated IR workflow framework

• Implement forensics data collection automation

• Research incident response best practices

Writing (4 hours):

• Blog Post: "Architecting Incident Response: Automation and Orchestration"

• Final conference talk polish

Community (2 hours):

• Null Space Labs engagement

Deliverables:

• Incident response framework

- Blog post published

Week 78: Conference Talk Delivery (14 hours)

Activities:

- Deliver conference talk (if accepted at local/virtual conference)
- Record and publish talk online
- Write accompanying blog post series
- Network with security community

Practice (6 hours):

- Final talk rehearsal
- Technical demo preparation
- Q&A; practice

Presentation (4 hours):

- Conference talk delivery
- Networking sessions

Writing (2 hours):

- Blog Post: Conference talk summary and key takeaways
- Share presentation materials

Community (2 hours):

- Conference participation

Deliverables:

- Conference talk delivered
- Presentation materials published
- Blog post published

Week 79: Thought Leadership Content (14 hours)

Activities:

- Write comprehensive technical whitepaper
- Create security architecture case studies
- Develop advanced tutorial series
- Build public speaking portfolio

Writing (10 hours):

- Technical whitepaper on advanced AppSec topic (20-30 pages)
- 2 case studies on real-world security architecture
- Advanced tutorial series (multi-part)

Research (2 hours):

- Identify future conference opportunities
- Research thought leadership platforms

Community (2 hours):

- OWASP LA meeting (present on whitepaper topic)

Deliverables:

- Technical whitepaper published
- 2 case studies published
- Tutorial series published

Week 80: Year 1 Review & Year 2 Planning (14 hours)

Activities:

- Comprehensive Year 1 review
- Update complete portfolio
- Plan Year 2 advanced specialization
- Prepare for principal engineer roles

Review (8 hours):

- Document all 52 weeks of learning
- Update GitHub repositories
- Polish all tools and frameworks
- Create comprehensive demo videos

Writing (4 hours):

- Year 1 comprehensive review blog post
- Year 2 learning roadmap article
- Update LinkedIn profile and resume

Community (2 hours):

- Year 1 celebration with OWASP LA community

Deliverables:

- Complete Year 1 portfolio
- Year 1 review blog post
- Year 2 roadmap

Quarter 4 Summary: Success Metrics

Completed by June 14, 2027:

- ■ Enterprise security architecture expertise
- ■ Advanced threat modeling framework
- ■ Zero trust architecture assessment capabilities
- ■ Security architecture review methodology
- ■ Secure-by-design pattern library
- ■ Microservices security architecture analyzer
- ■ Cloud-native security framework
- ■ Supply chain security architecture
- ■ Security metrics framework
- ■ Incident response automation
- ■ Conference talk delivered (thought leadership)
- ■ Technical whitepaper published
- ■ 13 technical blog posts published (total: 52 Year-to-Date)
- ■ Active community participation and thought leadership (26+ hours)

Complete Year 1 Summary: Final Success Metrics

Completed by June 14, 2027 (52 weeks after 28-week curriculum):

Technical Skills Mastered

- ■ OSWE Certification: Elite web application exploitation expertise
- ■ Cloud Security: AWS (IAM, S3, Lambda, CloudTrail), Kubernetes (RBAC, network policies, pod security, secrets), Infrastructure as Code security
- ■ Modern APIs: GraphQL, gRPC, REST, service mesh security (Istio/Linkerd), API gateway security
- ■ DevSecOps: CI/CD security, container security, supply chain security, secrets management (Vault)
- ■ Custom Tooling: SAST engine (multi-language), DAST framework, AI-powered testing, security orchestration platform
- ■ Enterprise Architecture: Security architecture review, threat modeling, zero trust, secure-by-design, microservices security, cloud-native security, incident response

Deliverables & Portfolio

- ■ 52+ Production Security Tools: OSWE exploitation toolkits, cloud security scanners, K8s security auditors, GraphQL/gRPC/REST API testing frameworks, CI/CD security gates, SAST/DAST engines, security orchestration platform, architecture review frameworks, and more
- ■ 52 Technical Blog Posts: Published on dev.to, establishing thought leadership and technical authority
- ■ Technical Whitepaper: 20-30 page advanced AppSec research paper
- ■ Conference Talk: Delivered at local or virtual security conference
- ■ Case Studies: 2+ real-world security architecture case studies
- ■ CVE Research: 2-4 CVE discoveries/submissions
- ■ Complete Portfolio Website: Professional showcase of all work

Community & Networking

- ■ 104+ Hours Community Engagement: Active OWASP LA and Null Space Labs participation
- ■ Thought Leadership: Conference speaking, whitepaper publication, advanced tutorials
- ■ Public Presence: Strong dev.to following, GitHub star growth, professional network expansion

Career Positioning

By June 14, 2027, you will be positioned for:

- Senior AppSec Engineer Roles: $165K-$185K salary range
- Principal Security Engineer / Security Architect Roles: $185K-$220K salary range
- Year 2 Advanced Specialization: Vulnerability research, academic papers, consulting practice
- P2P Project Leadership: Expansion and scaling as maintainer
- Consulting Practice Growth: $400K+ potential

Integration with 3-Year Elite AppSec Plan

This Year 1 reading schedule directly feeds into your 3-Year Elite AppSec Engineer Plan:

- Year 1 (This Plan): Technical depth → Senior AppSec Engineer level
- Year 2: Specialization + Research → Principal Engineer / Security Architect
- Year 3: Thought Leadership + Book → Global AppSec Authority

After completing this Year 1 plan, you'll be positioned for:

- Senior AppSec Engineer roles ($165K-$185K)
- Year 2 advanced learning (vulnerability research, academic papers, conference keynotes)
- P2P project expansion and scaling
- Consulting practice growth ($400K+ potential)

Resources Summary

Books Required (Approximate Cost: $800)

Q1 Books (OSWE + Cloud):

- The Web Application Hacker's Handbook ($50)
- Kubernetes Security ($45)
- Container Security ($40)
- AWS Security ($50)

Q2 Books (APIs + DevSecOps):

- GraphQL in Action ($50)
- Microservices Security in Action ($50)
- Securing DevOps ($45)

• The DevSecOps Playbook ($50)

Q3 Books (Tooling + Analysis):

• Engineering a Compiler ($90)
• The Art of Software Security Assessment ($75)
• Crafting Interpreters (Free online)
• The Tangled Web ($40)
• Real-World Bug Hunting ($40)
• Deep Learning for Coders ($50)

Q4 Books (Architecture + Thought Leadership):

• Security Engineering 3rd Edition (Free online)
• Threat Modeling: Designing for Security ($50)
• Zero Trust Networks ($45)
• Building Secure & Reliable Systems (Free online)
• Cloud Security and Privacy ($45)

Total Book Cost: ~$800

Online Resources (Free)

• OSWE Course: Included in certification purchase (~$1,600)
• Cloud Provider Documentation: AWS, Azure, GCP (Free)
• OWASP Resources: ASVS, SAMM, Top 10, Cheat Sheets (Free)
• Tool Documentation: Semgrep, CodeQL, Kubernetes, Istio (Free)
• CTF Platforms: flAWS, CloudGoat, Kubernetes Goat, DVGA (Free)

Community Memberships

• OWASP LA: Free
• Null Space Labs: ~$50/month membership (~$600/year)
• ISSA LA: ~$175/year membership (optional)

Total Estimated Cost: $1,400-$1,600 for complete Year 1 (post-28 week curriculum)

Weekly Schedule Template

Sustainable 18-Hour Week (Q1-Q3):

• Monday-Friday (Weekdays): Morning (2h) book reading, Afternoon (2h) hands-on practice, Evening (1h) labs or writing
• Saturday: Morning-Afternoon (4-5h) major project work, Evening (1-2h) community engagement
• Sunday: Morning (2-3h) review and planning, Afternoon: rest and recovery
• Total: 18 hours per week (sustainable long-term pace)

Reduced 14-Hour Week (Q4):

- Weekdays: 1.5-2 hours daily (focus on writing, architecture, thought leadership)
- Saturday: 3-4 hours project work
- Sunday: 2-3 hours review and planning
- Total: 14 hours per week (focusing on higher-level work)

Next Steps

1. Start Week 29 on June 16, 2026 (day after Week 28 completion)
2. Print this schedule and track progress weekly
3. Set up tracking spreadsheet for: hours logged, deliverables completed, CVEs discovered, blog posts published, conference talks submitted
4. Block calendar for weekly community engagement (OWASP LA 4th Wednesday monthly, Null Space Labs Tuesday open nights)
5. Order first batch of books for Q1 (weeks 29-41)
6. Set up OSWE course and lab environment immediately (Week 29 priority)
7. Prepare conference talk proposals (submit by Week 68-71)

Critical Reminders

REMEMBER: This is not just reading—it's systematic skill building toward elite AppSec technical authority.

Every week builds on the previous week. Every tool you build, blog post you write, CVE you discover, and conference talk you deliver compounds your credibility and expertise.

By June 14, 2027, you'll possess technical depth that rivals Senior AppSec Engineers with 10+ years of experience.

Your combination of:

- Intel enterprise threat modeling expertise (553+ documented threats)
- OSWE offensive security certification
- Cloud security mastery (AWS + Kubernetes)
- Custom SAST/DAST tooling
- Enterprise security architecture expertise
- Thought leadership presence (52 blog posts, conference talks, whitepaper)

...will position you uniquely in the AppSec market for Principal Engineer / Security Architect roles.

Sources:

- Post-26 Week 1-Year AppSec Reading Schedule from project knowledge (formatting structure, quarterly breakdown, resource organization, success metrics)

- Complete 28-Week AppSec Curriculum MERGED from project knowledge (curriculum completion date, skill foundation, tool portfolio baseline)
- Elite AppSec Engineer 3-Year Plan from project knowledge (Year 1 success metrics, quarterly objectives, career progression milestones)
- OWASP documentation, vendor resources, and academic research for technical content

Generated on: December 7, 2025

For: Fosres (Tanveer Salim)

Curriculum Type: Post-28 Week Year 1 Continuation

Duration: June 16, 2026 - June 14, 2027 (52 weeks)