Security Engineering Interview Prep Guide

Complementary Study Plan for General Security Engineering Roles

Generated: December 12, 2025

Duration: 28 weeks (parallel to AppSec curriculum)

Time Commitment: 8-12 hours/week (additional to AppSec study)

Target Roles: Security Engineer (General), Detection Engineer, Security Analyst

Based on: Grace Nolan's Google Security Engineering Interview Notes

Executive Summary

This guide provides a structured approach to preparing for general Security Engineering interviews alongside your focused 28-week AppSec curriculum. While your AppSec study emphasizes offensive security skills (vulnerability discovery, exploitation, secure code review), this complementary plan covers the broader defensive and infrastructure security topics commonly tested in Security Engineering interviews at companies like Google, Amazon, Microsoft, and other major tech firms.

Key Differentiator: Your AppSec curriculum already covers web application security, API security, and OWASP Top 10 in depth. This guide focuses on complementary domains: networking, OS internals, cryptography (beyond implementation), detection/monitoring, incident response, and security architecture.

How to Use This Guide

This study plan is designed to run in parallel with your 28-week AppSec curriculum. The topics are organized into 28 weekly modules that align with your existing schedule, requiring an additional 8-12 hours per week. Each week covers 2-3 related concepts with specific study materials and practical exercises.

Study Approach (from Grace Nolan's notes):

- Track concepts in three categories: 'To Learn', 'Revising', 'Done'
- Use spaced repetition - review terms regularly without immediately looking up answers
- Write one term per post-it note and move between categories as you master them
- Practice explaining concepts out loud, even to yourself
- Focus on understanding WHY security controls exist, not just memorizing definitions
- Apply knowledge through hands-on labs and coding challenges weekly

Interview Preparation Insights:

According to the Blind post you referenced, candidates who completed 85 Security Engineer on-sites emphasized that **coding ability is the #1 reason candidates fail Security Engineering interviews**. Your AppSec curriculum already addresses this with extensive Python development, but this guide adds security-specific coding challenges (log parsing, web scraping, port scanning, malware signature detection).

28-Week Study Plan Overview

Each week below maps to your existing AppSec curriculum week. Study these topics in parallel, dedicating 8-12 hours/week to this supplementary material.

Phase 1: Networking & Systems Foundation (Weeks 1-4)

Parallel to AppSec Phase 1: Python + SQL Injection Mastery

Week 1: TCP/IP Fundamentals & Network Protocols

• Study: OSI model layers, TCP 3-way handshake, UDP vs TCP, common port numbers (80, 443, 22, 3389)

• Study: IP addressing (IPv4/IPv6), subnetting, CIDR notation, NAT concepts

• Reading: Computer Networking: A Top-Down Approach (Ch 1-3) or free alternative: beej.us/guide/bgnet/

• Lab: Wireshark packet capture analysis - capture HTTP, HTTPS, DNS traffic and analyze packet structure

• Coding Challenge: Write a simple port scanner in Python using socket library

Week 2: DNS, TLS/SSL, and Certificate Management

• Study: DNS resolution process, DNS record types (A, AAAA, MX, CNAME, TXT), DNS security (DNSSEC)

• Study: TLS handshake process, certificate chains, certificate transparency logs

• Study: Common TLS attacks (Heartbleed, POODLE, BEAST) and mitigations

• Reading: High Performance Browser Networking by Ilya Grigorik (free online)

• Lab: Use openssl to inspect certificates, analyze certificate chains, verify certificate validity

• Coding Challenge: Build a certificate expiration checker that monitors domains

Week 3: Firewalls, VPNs, and Network Segmentation

• Study: Firewall types (packet filtering, stateful, application layer), iptables basics

• Study: VPN protocols (IPsec, OpenVPN, WireGuard), site-to-site vs remote access

• Study: Network segmentation strategies, VLANs, DMZ architecture

• Study: Zero Trust networking principles

• Reading: NIST SP 800-41r1 (Firewall Guidelines) and NIST SP 800-77 (VPN Guidelines)

• Lab: Configure iptables rules on Linux VM, test firewall behavior with nmap

• Practical: Design network architecture for a fictional company with proper segmentation

Week 4: Linux OS Internals & System Calls

• Study: Linux file system hierarchy, file permissions (chmod, chown, ACLs)

• Study: Process management (ps, top, /proc filesystem), signals, process lifecycle

• Study: System calls (open, read, write, fork, exec), strace for debugging

• Study: Linux security modules (SELinux, AppArmor), capabilities

• Reading: The Linux Programming Interface by Michael Kerrisk (Ch 1-5) or free Linux documentation

• Lab: Use strace to analyze program behavior, identify suspicious system calls

• Coding Challenge: Write a process monitor that detects when new processes spawn

Phase 2: Cryptography & Binary Exploitation (Weeks 5-7)

Parallel to AppSec Phase 2: XSS, CSRF, Access Control

Week 5: Cryptography Fundamentals Beyond Implementation

• Study: Symmetric encryption (AES modes - CBC, GCM, CTR), authenticated encryption (AEAD)

• Study: Asymmetric encryption (RSA, ECC), key exchange (Diffie-Hellman, ECDH)

• Study: Hash functions (SHA-256, SHA-3), HMAC, password hashing (bcrypt, Argon2)

• Study: Common cryptographic vulnerabilities (padding oracle, timing attacks, IV reuse)

• Reading: Serious Cryptography by Jean-Philippe Aumasson (Ch 1-8)

• Lab: CryptoHack challenges (cryptohack.org) - complete 10-15 beginner challenges

• Practical: Explain difference between GCM and CBC modes, when to use each

Week 6: Memory Corruption & Binary Exploitation Basics

• Study: Stack layout, buffer overflows, stack canaries, NX bit, ASLR

• Study: Heap exploitation basics, use-after-free vulnerabilities

• Study: Format string vulnerabilities

• Study: Modern mitigations (DEP, CFI, SafeStack)

• Reading: Hacking: The Art of Exploitation by Jon Erickson (Ch 3-5)

• Lab: picoCTF binary exploitation challenges or Pwnable.kr (10 challenges)

• Practical: Write a vulnerable C program and exploit it in a controlled environment

Week 7: Windows Security Architecture

• Study: Windows authentication (Kerberos, NTLM), Active Directory fundamentals

• Study: Windows access control (ACLs, security descriptors, tokens)

• Study: Windows security features (AMSI, Credential Guard, Windows Defender)

• Study: Common Windows attacks (pass-the-hash, golden tickets, Mimikatz)

• Reading: Windows Internals by Russinovich (Ch 6-7) or Microsoft Windows Security documentation

• Lab: Set up Windows VM, explore Active Directory, practice basic AD enumeration

• Practical: Explain how Kerberos authentication works in Active Directory

Phase 3: Detection, Monitoring & Incident Response (Weeks 8-10)

Parallel to AppSec Phase 3: Authentication & Authorization

Week 8: Log Analysis & SIEM Fundamentals

• Study: Common log sources (syslog, Windows Event Logs, web server logs, firewall logs)

• Study: Log aggregation, parsing, normalization strategies

• Study: SIEM basics (Splunk, ELK stack), detection rules and alerts

• Study: Key indicators of compromise (IOCs) in logs

• Reading: Applied Security Visualization by Raffael Marty (Ch 1-4)

• Lab: Install ELK stack locally, ingest sample logs, create detection rules

• Coding Challenge: Write a log parser that extracts IOCs (IPs, domains, file hashes)

Week 9: Network Intrusion Detection & Traffic Analysis

• Study: IDS vs IPS, signature-based vs anomaly-based detection

• Study: Snort/Suricata rule syntax, common attack patterns in network traffic

• Study: Network forensics, PCAP analysis techniques

• Study: Common network-based attacks (port scans, DDoS, C2 beaconing)

• Reading: The Practice of Network Security Monitoring by Richard Bejtlich

• Lab: Analyze malicious PCAP files from malware-traffic-analysis.net (5-10 samples)

• Practical: Write Suricata rules to detect common attacks (SQL injection in HTTP, SSH brute force)

Week 10: Incident Response & Digital Forensics Basics

• Study: Incident response lifecycle (NIST SP 800-61), incident classification

• Study: Evidence collection and preservation, chain of custody

• Study: Disk forensics basics (file systems, deleted file recovery, timeline analysis)

• Study: Memory forensics (Volatility framework), process analysis

• Reading: SANS Incident Handler's Handbook (free PDF from SANS)

• Lab: Analyze a compromised system image using Autopsy or similar forensics tool

• Practical: Create an incident response playbook for web application compromise

Phase 4: Advanced Security Topics (Weeks 11-13)

Parallel to AppSec Phase 4: SAST/DAST & Security Automation

Week 11: Malware Analysis Fundamentals

• Study: Malware types (viruses, worms, trojans, ransomware, rootkits)

• Study: Static analysis techniques (strings, PE structure, imports/exports)

• Study: Dynamic analysis (sandbox analysis, behavior monitoring)

• Study: Common malware persistence mechanisms, anti-analysis techniques

• Reading: Practical Malware Analysis by Sikorski & Honig (Ch 1-7)

• Lab: Analyze 5-10 malware samples using tools (PEStudio, Ghidra, Wireshark)

• Coding Challenge: Write a malware signature scanner using YARA rules

Week 12: Reverse Engineering Basics

• Study: x86/x64 assembly basics, common instructions (mov, push, pop, jmp, call)

• Study: Function calling conventions (cdecl, stdcall), stack frames

• Study: Disassemblers (IDA Pro, Ghidra), debuggers (gdb, x64dbg)

• Study: Code obfuscation techniques, packing/unpacking

• Reading: Reverse Engineering for Beginners by Dennis Yurichev (free online)

• Lab: Reverse engineer crackme challenges from crackmes.one (5-10 beginner challenges)

• Practical: Reverse engineer a simple keygenme to understand the algorithm

Week 13: Threat Modeling & Security Architecture

• Study: STRIDE threat modeling framework (Spoofing, Tampering, Repudiation, Info Disclosure, DoS, Elevation)

• Study: Attack trees, data flow diagrams for threat modeling

• Study: Security architecture patterns (defense in depth, least privilege, separation of duties)

• Study: Secure design principles (fail securely, secure defaults, minimize attack surface)

• Reading: Threat Modeling: Designing for Security by Adam Shostack

• Practical: Create STRIDE threat model for a fictional e-commerce system

• Review: Leverage your Intel threat modeling experience - this is a major differentiator!

Phase 5: Cloud Security & Infrastructure (Weeks 14-16)

Parallel to AppSec Phase 5: API Security Research & Phase 6: Production Tools

Week 14: Container Security (Docker & Kubernetes)

• Study: Docker security (image scanning, runtime security, seccomp profiles)

• Study: Kubernetes security architecture (RBAC, network policies, pod security standards)

• Study: Container breakout techniques, privilege escalation in containers

• Study: Container security tools (Trivy, Clair, Falco)

• Reading: Container Security by Liz Rice

• Lab: Complete Kubernetes Goat scenarios (madhuakula.com/kubernetes-goat/)

• Practical: Configure secure Kubernetes RBAC policies for multi-tenant cluster

Week 15: AWS Security Deep Dive

• Study: AWS IAM best practices, policy evaluation logic, cross-account access

• Study: AWS security services (GuardDuty, Security Hub, CloudTrail, Config)

• Study: S3 security (bucket policies, ACLs, public access blocks), S3 data exposure risks

• Study: Lambda security, API Gateway security, secrets management (Secrets Manager/SSM)

• Reading: AWS Security Best Practices whitepaper

• Lab: Complete flAWS challenge (flaws.cloud and flaws2.cloud) - all 6 levels

• Practical: Audit an AWS account using Scout Suite, identify misconfigurations

Week 16: Infrastructure as Code Security

• Study: Terraform security best practices, state file security

• Study: CloudFormation security, detecting insecure configurations

• Study: Secrets management in IaC (HashiCorp Vault, git-secrets)

• Study: Policy-as-code tools (OPA, Sentinel)

• Reading: Infrastructure as Code Security by Syed & Calavera

• Lab: Use Checkov to scan IaC for security issues, fix identified problems

• Practical: Write OPA policies to enforce security guardrails in Terraform

Phase 6-10: Advanced Preparation & Interview Practice (Weeks 17-28)

Note: Weeks 17-28 focus on consolidating knowledge, mock interviews, and targeted practice based on specific company interview patterns. These weeks should be adapted based on which companies you're actively interviewing with.

Week 17-19: Security-Specific Coding Challenges

• Practice coding challenges from Grace Nolan's notes:

- Text parsing & manipulation (cypher implementation, string encoding)
- Log parsing (extract domains, IPs, timestamps, executable names)
- Web scraping for security data collection
- Port scanner implementation
- Credential generator and login tracker
- PDF metadata forensics tool
- Malware signature detection in binaries

• Complete 20-30 LeetCode Easy/Medium problems with security focus

• Build 3-5 complete security tools demonstrating coding proficiency

Week 20-22: Web Application Security Depth

• Study: OWASP Top 10 from defensive/detection perspective (complementary to your AppSec offensive skills)

• Study: Web application firewalls (WAF), ModSecurity rule writing

• Study: Content Security Policy (CSP), HTTP security headers

• Study: API security beyond exploitation - rate limiting, authentication schemes

• Lab: Deploy and configure ModSecurity, write custom WAF rules

• Practical: Design defense-in-depth web application security architecture

Week 23-25: Mock Interviews & System Design

• Practice security system design interviews:

- Design a secure authentication system
- Design a SIEM architecture for enterprise
- Design network segmentation for cloud environment
- Design incident response workflow and tooling

• Conduct mock interviews with peers or mentors

• Record yourself explaining security concepts, review for clarity

• Practice whiteboarding threat models and architecture diagrams

Week 26-28: Company-Specific Preparation & Review

• Research specific interview patterns for target companies:

- Google: Heavy focus on coding, system design, deep technical knowledge

- Amazon: Leadership principles, behavioral questions, threat modeling

- Microsoft: Windows security depth, cryptography, security architecture

• Review all spaced-repetition cards, ensure mastery of core concepts

• Complete final practice problems in weak areas

• Prepare thoughtful questions to ask interviewers about security team structure and priorities

Recommended Resources

These resources are curated based on Grace Nolan's notes and the Blind post recommendations. Focus on free resources first, then invest in paid materials for areas where you need depth.

Essential Free Resources:

- Grace Nolan's Security Engineering Interview Notes (github.com/gracenolan/Notes)
- SANS Reading Room (free whitepapers on all security topics)
- NIST Cybersecurity Publications (SP 800 series)
- High Performance Browser Networking by Ilya Grigorik (hpbn.co)
- Beej's Guide to Network Programming (beej.us/guide/bgnet/)
- CryptoHack (cryptohack.org) - interactive cryptography challenges
- PicoCTF (picoctf.org) - beginner-friendly CTF challenges
- Pwnable.kr - binary exploitation practice
- Malware Traffic Analysis (malware-traffic-analysis.net)
- Reverse Engineering for Beginners by Dennis Yurichev (free PDF)

Essential Paid Books (~$500 total investment):

- Computer Networking: A Top-Down Approach by Kurose & Ross (~$80)
- The Linux Programming Interface by Michael Kerrisk (~$70)
- Serious Cryptography by Jean-Philippe Aumasson (~$40)
- Hacking: The Art of Exploitation by Jon Erickson (~$40)
- Practical Malware Analysis by Sikorski & Honig (~$50)
- The Practice of Network Security Monitoring by Richard Bejtlich (~$45)
- Threat Modeling: Designing for Security by Adam Shostack (~$50)
- Container Security by Liz Rice (~$40)
- Applied Security Visualization by Raffael Marty (~$55)

Hands-On Labs & Platforms:

- TryHackMe (tryhackme.com) - $10/month for guided security learning paths
- HackTheBox (hackthebox.com) - Free tier available, $14/month for VIP
- Pentester Academy (pentesteracademy.com) - Various courses, $49-199/month
- LetsDefend (letsdefend.io) - Blue team simulation platform
- Crackmes.one - reverse engineering challenges (free)

- flAWS and flAWS2 - AWS security challenges (free)
- Kubernetes Goat - Kubernetes security scenarios (free)

Weekly Time Allocation Template

Example 10-hour week breakdown to complement your 24-hour AppSec study schedule. Adjust based on your learning pace and schedule constraints.

Day	Time Block	Activity	Hours
Monday	Morning	Read assigned chapter/documentation	2h
Tuesday	Evening	Complete labs/hands-on exercises	2h
Wednesday	Morning	Practice coding challenges	1.5h
Thursday	Evening	Review concepts with spaced repetition	1h
Friday	Morning	Complete remaining labs/challenges	1.5h
Saturday	Afternoon	Deep dive on weak areas, research	2h
Sunday	Evening	Weekly review, update tracking system	1h
		<b>Total Weekly Time</b>	**<b>11h</b>**

Key Principle: Consistency beats intensity. 10-12 hours per week consistently over 28 weeks will build deeper knowledge than sporadic 20-hour cramming sessions. Protect your study time and maintain the schedule even when it feels difficult.

Security Engineering Interview Tips

These insights are synthesized from Grace Nolan's notes and feedback from the Blind post author who completed 85 Security Engineer interviews at top tech companies.

General Interview Approach:

- Questions are intentionally vague to encourage clarifying questions - always ask for clarification
- Write down notes about the question to avoid forgetting details or partially answering
- Repeat the question back to interviewer to confirm understanding and buy thinking time
- Think out loud - interviewers evaluate your problem-solving process, not just final answers
- Interviews should feel like collaborative conversations with back-and-forth
- Don't jump too quickly to solutions - thoroughly explore scenarios first

Technical Depth Expectations:

- Expect questions that push technical knowledge to your limits - this is intentional
- When you don't know something, explain your thinking process and what you'd do to find the answer
- Interviewers look for how deeply you understand topics, not memorized definitions
- Approach questions from low-level details before moving to high-level abstractions
- If given feedback that assumptions are unreasonable, adjust gracefully

Coding Interview Specifics:

- **CRITICAL:** Coding ability is the #1 reason candidates fail Security Engineering interviews
- Practice security-specific coding challenges: log parsers, web crawlers, port scanners
- Be prepared for LeetCode Medium-level problems in addition to security-specific challenges
- Write clean, readable code with proper error handling and edge case consideration
- Explain trade-offs in your implementation choices (time/space complexity, security implications)
- Test your code with example inputs before declaring it complete

Demonstrating Breadth vs Depth:

- Some questions test role-specific depth, others test breadth of security knowledge
- When asked about unfamiliar topics, show willingness to explore and make educated guesses
- Connect new concepts to things you do know to demonstrate learning ability
- It's okay to say 'I don't know but here's how I'd approach learning this'

Mindset & Perspective:

• Interviews are NOT a measure of you being 'good enough' - they're imperfect evaluation snapshots

• Companies love to see how candidates have grown skills over time - reapplying is normal

• If you don't get an offer, identify weak areas, improve them, and try again

• Interviewers can only evaluate what you say - speak your thoughts clearly

Leveraging Your Intel Experience

Your background provides unique advantages in Security Engineering interviews. Here's how to effectively communicate your differentiating experience:

Threat Modeling at Scale (553+ Threats Documented):

- Most candidates have either offensive OR defensive skills, rarely both at enterprise scale
- Emphasize: 'I documented 65.83% of Intel's entire Threat Modeling Database using STRIDE methodology'
- Highlight template creation: '100+ engineers use my reusable threat model templates'
- Connect to Security Engineering: 'I can identify threats at the architecture stage, not just after deployment'

Systems Programming & Cryptography Background:

- C/C++/Golang experience creates premium positioning for memory safety vulnerabilities
- Cryptographic implementations (XMSS, ChaCha20, Argon2) demonstrate deep understanding beyond API usage
- Intel Crypto Academy Level I training shows commitment to cryptographic best practices
- Most AppSec candidates focus on web vulnerabilities - you bring binary analysis depth

Communicating Your Transition Story:

- Frame transition positively: 'Moving from design-stage security to runtime security engineering'
- Emphasize complementary skills: 'My threat modeling helps me think like an attacker while building defenses'
- Demonstrate initiative: 'I'm investing 24 hours/week mastering AppSec through systematic self-study'
- Show results: 'I've completed 85+ PortSwigger labs and built 20+ security tools in my curriculum'

Progress Tracking System

Use this structured approach to track your progress through both curriculums. The tracking system ensures accountability and helps identify areas needing more focus.

Weekly Review Template:

1. Concepts Mastered: List 5-10 new terms/concepts you can now explain clearly
2. Labs Completed: Document which hands-on exercises you finished
3. Coding Challenges: Track security coding problems solved this week
4. Weak Areas Identified: Note topics requiring additional study
5. Interview Readiness: Rate 1-10 for each topic area studied
6. Time Invested: Log actual hours spent (goal: 10-12 hours/week)

Spaced Repetition System (from Grace Nolan):

- Create post-it notes for each concept: 'To Learn', 'Revising', 'Done'
- Move concepts between categories as you master them
- Review 'Revising' concepts every 2-3 days without looking up answers first
- Carry a notebook everywhere - write down terms and explanations
- Practice explaining concepts out loud, even to yourself
- Use falling asleep time to mentally review terms (Grace guarantees <10 min to sleep!)

Monthly Milestone Checkpoints:

- Month 1 (Weeks 1-4): Networking fundamentals, TCP/IP, DNS, TLS mastery
- Month 2 (Weeks 5-8): Cryptography depth, binary exploitation basics, log analysis
- Month 3 (Weeks 9-12): Network forensics, incident response, malware analysis
- Month 4 (Weeks 13-16): Threat modeling, container security, cloud security
- Month 5-6 (Weeks 17-24): Coding challenges, system design, interview practice
- Month 7 (Weeks 25-28): Company-specific prep, mock interviews, final review

Integration with Your AppSec Curriculum

This Security Engineering prep guide is designed to complement, not replace, your comprehensive 28-week AppSec curriculum. Here's how the two plans work together:

Synergies Between Plans:

- AppSec curriculum provides offensive security depth (OWASP Top 10, exploitation, secure code review)
- Security Engineering plan adds defensive depth (detection, incident response, architecture)
- Your Python coding practice in AppSec directly supports Security Engineering coding challenges
- Cloud security (Week 20-21 AppSec) aligns with AWS/K8s security (Week 14-16 this plan)
- STRIDE threat modeling practice (Week 13 this plan) reinforces your Intel experience

Time Management Strategy:

- AppSec curriculum: 24 hours/week (primary focus)
- Security Engineering prep: 10-12 hours/week (complementary focus)
- Total investment: 34-36 hours/week across both plans
- Sustainable approach: If overwhelmed, prioritize AppSec curriculum and scale back Security Engineering to 6-8 hours/week
- Flexible adaptation: Some weeks allow more time for Security Engineering (e.g., portfolio week)

When to Prioritize Each Plan:

- Prioritize AppSec for: Application Security Engineer roles (Trail of Bits, NCC Group, GitLab)
- Prioritize Security Engineering for: General Security Engineer roles (Google, Amazon, Microsoft)
- Balanced approach for: Companies that value both (Stripe, Coinbase, Anthropic)
- Use resume/job descriptions to guide focus: Match study time to job requirements

Company-Specific Interview Patterns

Based on Grace Nolan's notes and community feedback, here are known patterns for major tech companies. Use this to prioritize your preparation in weeks 26-28.

Google Security Engineering:

- Heavy emphasis on coding - expect LeetCode Medium level problems
- Deep technical dives on networking, cryptography, and system internals
- Security system design questions (design a SIEM, secure authentication, etc.)
- Behavioral questions using 'Googleyness' framework
- Grace Nolan's notes are specifically optimized for Google interviews

Amazon Security Engineering:

- Leadership principles feature heavily in behavioral questions
- Threat modeling and security architecture design
- AWS security knowledge is strong advantage but not always required
- Coding challenges often security-focused (log parsers, web crawlers)

Microsoft Security:

- Windows security internals knowledge valued highly
- Cryptography depth, especially around TLS/authentication protocols
- Security architecture for enterprise-scale systems
- Threat modeling for Microsoft products/services

Startup/Mid-Size Companies (Trail of Bits, NCC Group):

- More emphasis on practical exploitation skills and tool development
- Code review exercises, vulnerability research methodology
- Your AppSec curriculum directly aligns with these companies
- Client communication skills and consulting mindset important

Final Advice & Encouragement

You're embarking on an ambitious dual-curriculum study plan totaling 34-36 hours per week for 28 weeks. This level of commitment is exceptional and will position you strongly for Security Engineering roles.

Key Success Factors:

1. **Consistency Over Intensity:** 10 hours every week beats 40 hours one week then nothing
2. **Active Learning:** Labs and coding challenges beat passive reading every time
3. **Spaced Repetition:** Review concepts regularly, don't cram
4. **Teach to Learn:** Explain concepts to others (or yourself) to identify gaps
5. **Build in Public:** Document your learning journey on GitHub and dev.to
6. **Community Engagement:** OWASP LA meetings and Null Space Labs provide accountability
7. **Adaptive Planning:** Adjust based on job opportunities and interview feedback

When You Feel Overwhelmed:

- Remember: You don't need to master everything in this guide to get interviews
- Prioritize breadth over depth initially - you can go deep after landing a role
- Your Intel threat modeling experience already differentiates you significantly
- Scale back to 6-8 hours/week on this plan if needed - AppSec curriculum takes priority
- Take breaks when needed - burnout helps no one

Your Competitive Advantages:

- Enterprise-scale threat modeling experience (553+ threats) is rare and valuable
- Systems programming background (C/C++/Golang) differentiates you from web-focused candidates
- Cryptography implementation experience shows depth beyond typical candidates
- Dual curriculum approach demonstrates exceptional commitment and work ethic
- You're building both offensive AND defensive security skills - most candidates have only one

Remember Grace Nolan's journey: She went from career confusion to Google Security Engineer through intensive study and persistence. The Blind post author completed 85 interviews before succeeding. This is a marathon, not a sprint. Your systematic approach and diverse background position you well for success. Trust the process, stay consistent, and good luck!

Quick Reference Checklists

Week 1 Checklist Example:

- ■ Read networking chapters (OSI model, TCP/IP, UDP)
- ■ Complete Wireshark packet analysis lab
- ■ Write port scanner in Python
- ■ Create spaced-repetition cards for 10+ networking terms
- ■ Practice explaining TCP 3-way handshake out loud
- ■ Document learning in weekly review template

Pre-Interview Checklist:

- ■ Review all spaced-repetition cards - can you explain each concept?
- ■ Complete 5-10 security coding challenges in interview conditions
- ■ Practice whiteboarding system design problems
- ■ Review company-specific patterns and focus areas
- ■ Prepare STAR method stories highlighting Intel threat modeling experience
- ■ Prepare thoughtful questions for interviewers about team and technology
- ■ Test technical setup (video, screensharing, coding environment)

Monthly Self-Assessment:

Rate yourself 1-10 on these dimensions after each month:

- ■ Networking fundamentals (TCP/IP, DNS, TLS)
- ■ Operating systems internals (Linux, Windows)
- ■ Cryptography (beyond implementation - understanding modes, attacks)
- ■ Detection and monitoring (SIEM, log analysis, IDS)
- ■ Incident response and forensics
- ■ Cloud security (AWS, Kubernetes, containers)
- ■ Malware analysis and reverse engineering
- ■ Security architecture and threat modeling
- ■ Coding ability (security-specific challenges)
- ■ Interview communication skills

Appendix: Additional Resources & References

Primary Source Documents:

- Grace Nolan's Security Engineering Interview Notes: github.com/gracenolan/Notes/blob/master/interview-study-notes-for-security-engineering.md
- Blind Post: 'I did 85 security engineer on-sites with top tech companies - a prep guide' (teamblind.com)
- NIST Cybersecurity Framework: nist.gov/cyberframework
- OWASP Top 10: owasp.org/www-project-top-ten/

Community & Networking:

- OWASP LA Chapter: Monthly meetings (4th Wednesday)
- Null Space Labs: Tuesday open nights in Burbank
- r/netsec Reddit community for security news and discussions
- Security Twitter: Follow @GraceNolan_, @SwiftOnSecurity, @threatintel for daily insights

YouTube Channels for Visual Learners:

- LiveOverflow - Binary exploitation and CTF walkthroughs
- IppSec - HackTheBox machine walkthroughs
- John Hammond - CTF challenges and security tutorials
- PwnFunction - Animated security concept explanations
- Computerphile - Computer science and security fundamentals

Certifications to Consider (Optional):

- OSCP (Offensive Security Certified Professional) - Strong for offensive skills
- GIAC GCIA (Certified Intrusion Analyst) - Strong for detection/monitoring
- AWS Certified Security - Specialty - Strong for cloud security roles
- CompTIA Security+ - Good foundation but not required for experienced candidates

Note: Your experience and demonstrated skills through projects matter more than certifications

Document Version: 1.0
Last Updated: December 12, 2025
Created for: Tanveer Salim (fosres)
Purpose: Complementary Security Engineering interview prep alongside 28-week AppSec curriculum