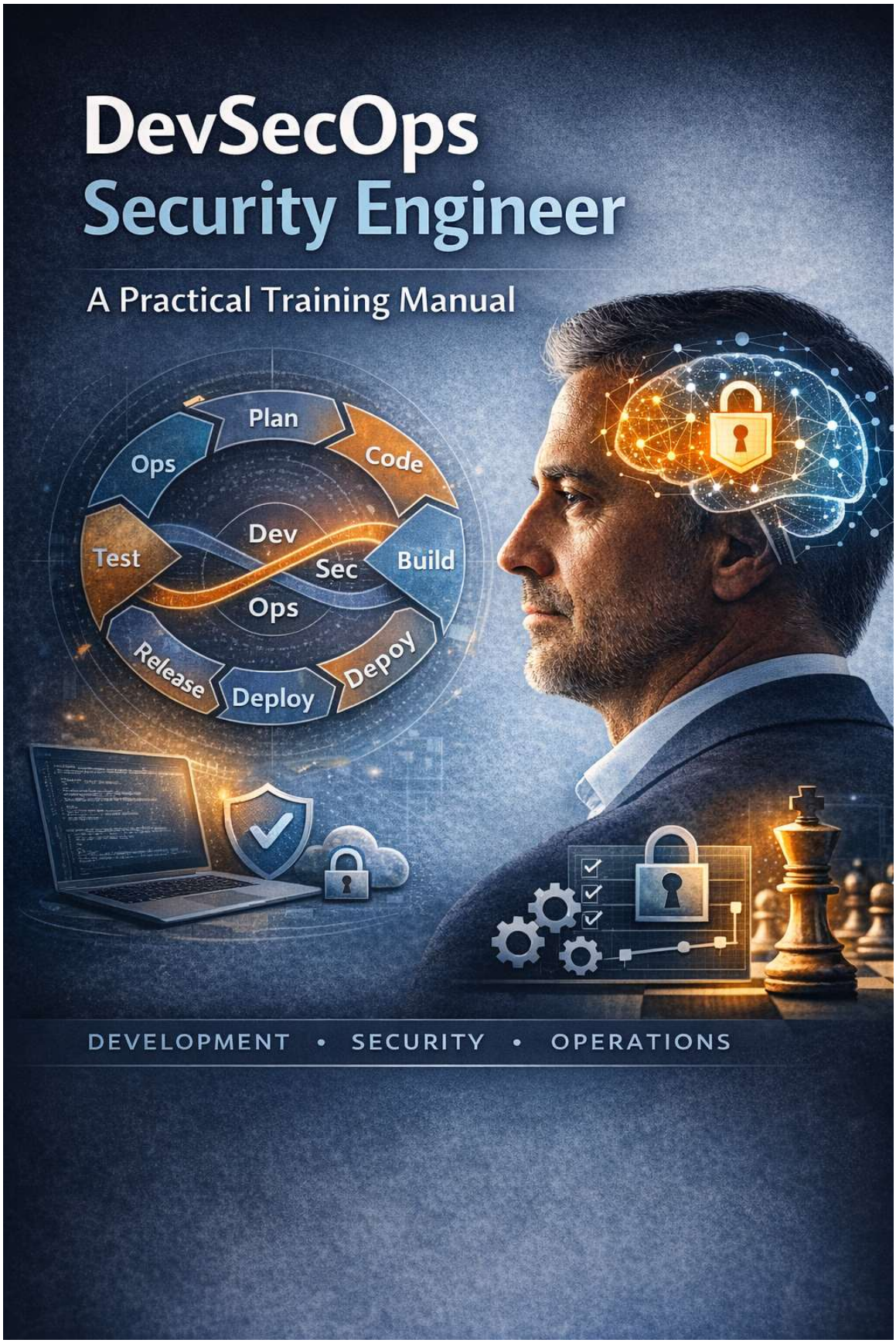
www.TerminuSys.com



Dr. Nader Iranpour
niranpour@gmail.com

DevSecOps Security Engineer

A Practical Training Manual

1. Why DevOps Exists

Consider a public organization such as ABC building a web application called **"Claims Portal"**—a system that allows injured workers to submit claims, upload medical documents, and track claim status.

The organization needs:

- Fast delivery
- High reliability
- Strong security
- Frequent updates without downtime

These goals are often in tension. Traditional software delivery models struggled to achieve all of them simultaneously. This is the problem **DevOps** was created to solve.

Before DevOps (The Old World)

Historically:

- Developers wrote code
- Security reviewed it at the end
- Operations deployed it
- Teams worked in silos
- Releases took months
- Security was perceived as a blocker

DevOps emerged to eliminate these bottlenecks by changing **how teams work together**, not just which tools they use.

2. What DevOps Really Means

DevOps is a way of working where development and operations collaborate to deliver software quickly, safely, and reliably through automation and shared responsibility.

Core ideas include:

- Small, incremental changes
- Frequent releases
- Automation over manual work

Dr. Nader Iranpour
niranpour@gmail.com

- Continuous feedback loops
- Shared ownership of outcomes

DevOps is not CI/CD alone. CI/CD is an *implementation mechanism* of DevOps thinking.

3. The SDLC in a DevOps World

The Software Development Life Cycle still exists, but it operates differently:

1. Plan
2. Design
3. Develop
4. Build (Continuous Integration)
5. Test
6. Release (Package)
7. Deploy (Continuous Delivery)
8. Operate
9. Monitor and Improve

Instead of linear handoffs, these stages are **continuous and interconnected**.

4. How DevOps Works — A Step-by-Step Story

Planning

A product owner defines a small feature:

"Allow users to upload medical documents."

This becomes a **user story** with clear acceptance criteria and is added to the backlog. DevOps favors **small, well-defined changes** over large releases.

Design

Architects design:

- Web frontend
- Backend APIs
- Database
- Cloud infrastructure (e.g., Azure)

They define data flows, service interactions, and dependencies.

This is where security must begin, not later.

Development

Developers work in feature branches, commit code to Git, and push small, frequent changes. Each change is traceable and reviewable.

Continuous Integration (CI)

Every code change automatically triggers:

- Compilation
- Unit tests
- Code quality checks
- Artifact creation

Failures stop the pipeline immediately—no manual intervention required.

Testing

All testing is automated:

- Unit tests
- Integration tests
- API tests
- UI tests

Automation ensures speed and consistency.

Build & Package

Applications are packaged as immutable artifacts (e.g., Docker images). Each artifact is versioned and ready for deployment.

Continuous Delivery / Deployment (CD)

Artifacts are deployed through environments (Dev, Test, Staging, Production) either with approvals or fully automated, depending on maturity.

Operations

The application runs in production. Users submit claims. Operations teams ensure availability, reliability, and scalability.

Monitoring & Feedback

Logs, metrics, and alerts feed back into development and planning. DevOps thrives on **learning loops**.

5. CI/CD in One Sentence

A **CI/CD pipeline** is an automated assembly line that safely and repeatedly moves code from a developer's laptop to production.

6. Introducing DevSecOps

DevSecOps means embedding security into every DevOps stage *without slowing it down*.

Security becomes:

- A design input
- A coding guide
- An automated test
- A release gate
- A monitoring signal

Security shifts from **inspection** to **enablement**.

7. Security Responsibilities Across the DevOps Lifecycle

Planning Stage

Security focuses on **early risk identification**.

Key activities:

- Identify sensitive data (e.g., medical records)
- Identify regulatory requirements
- Define security acceptance criteria

Security requirements are added directly into user stories.

Design Stage

Security prevents major flaws before code exists.

Key activities:

- Threat modeling using STRIDE
- Secure architecture review
- Validation of authentication, authorization, encryption, segmentation, and secrets management

Outputs are actionable mitigations—not reports.

Development Stage

Security prevents insecure code from being written.

Key activities:

- Secure coding guidance
- Manual secure code reviews
- Static Application Security Testing (SAST)

Focus is on logic, authorization, input validation, and secrets handling.

Build Stage (CI)

Security becomes automated.

Key activities:

- Software Composition Analysis (SCA)
- Security quality gates
- False-positive tuning

Pipelines fail only for **meaningful risk**, not noise.

Testing Stage

Security tests running applications.

Key activity:

- Dynamic Application Security Testing (DAST)

This identifies runtime flaws and misconfigurations.

Pre-Release Stage

Security provides deep assurance.

Key activity:

- Manual penetration testing (black-box and white-box)

Focus is on business logic and chained attacks.

Deployment Stage

Security enforces safe infrastructure.

Key activities:

- Cloud security policies
- Secure baselines via Infrastructure as Code

Operations Stage

Security detects and responds.

Key activities:

- Monitoring logs and alerts
- Identifying suspicious behavior
- Supporting incident response

Incident Response & Feedback

Security supports:

- Containment and investigation
- Root cause analysis
- Continuous improvement

Lessons learned feed back into threat models, pipelines, and standards.

8. What a DevSecOps Security Engineer Actually Does

A DevSecOps Security Engineer:

- Thinks in **systems**, not tools
- Enables developers rather than blocking them
- Embeds security into pipelines and architecture
- Focuses on risk, not checklists
- Treats security as a continuous process

Security success is measured not by the number of controls, but by **how early and effectively risk is reduced**.

9. Core Insight for Learners

- DevOps is not just CI/CD.
- Security is not a final gate.
- DevSecOps is not about more tools.

It is about better decisions, earlier.

www.TerminuSys.com

Dr. Nader Iranpour
niranpour@gmail.com