HORIZONSHIELD

Securing Your Digital Future, Today.

APPSEC SERIES · BOOK 08 OF 16

DevSecOps Implementation Guide

Shifting Security Left — 2026 Playbook

Integrate security throughout the software development lifecycle. Secrets management, SAST, DAST, SCA, container security, supply chain security, and security as code — with GitHub Actions pipeline examples and tool recommendations.

PIPELINE STAGES	TOOLS	STANDARD	LEVEL
8	60+	OWASP SAMM	Intermediate

Contents

DEVSECOPS IMPLEMENTATION 2026

CHAPTER 01

DevSecOps Culture & Shift-Left Security

Security is everyone's responsibility. Shifting security left reduces the cost of fixing vulnerabilities by 100x.

SHIFT LEFT | SECURITY CHAMPIONS | OWASP SAMM | CULTURE

Shift-Left Security Economics

Phase	Traditional Security	DevSecOps Approach	Cost to Fix Vulnerability
Design	Post-development architecture review	Threat modelling, security requirements review	1x — cheapest point to fix
Development	Manual code review (if at all)	SAST in IDE + pre-commit hooks with instant feedback	6x — code written but not merged
Build/CI	No scanning	SAST, secrets detection, dependency scan — block on critical	15x — build pipeline catch
Test	Manual pentest once per release	DAST, API fuzzing, IAST, automated security regression	45x — QA environment fix
Staging	Nothing	Full DAST + compliance checks + IaC scanning	100x — staging environment
Production	Emergency patch after incident	Runtime protection + monitoring + rapid response	300x — live system exploitation

Security Champions Programme

1 **Identify Champions**

MONTH 1

Identify one security-interested developer in each team. Security champions are NOT security professionals — they are developers with security interest who bridge the gap between dev and security teams.

---

2 **Train & Equip**

MONTH 2-3

Security champions training: OWASP Top 10, secure coding practices, threat modelling basics, tool usage (SAST, dependency scanning). Budget for SANS or equivalent security training.

---

3 **Integrate into Process**

MONTH 3-4

Champions participate in: security review of design documents, code review for security issues, security tool configuration for their team, security findings triage and prioritisation.

---

4 **Measure & Reward**

ONGOING

Track security debt reduction per team. Champions presented at leadership meetings. Security metrics included in team OKRs. Recognition in performance reviews.

> **THE COST OF FINDING VULNERABILITIES EARLY**
>
> IBM SAST: vulnerabilities found during coding cost $80 to fix. Found during testing: $960. Found in production: $7,600. Found after a breach: $240,000+ (breach cost + remediation + regulatory fines). Every $1 invested in shift-left security saves $30-100 in remediation costs.

CHAPTER 02

Secrets Management

The most dangerous commit you will ever make is one containing a credential.

SECRETS DETECTION | VAULT | GITLEAKS | ENVIRONMENT VARIABLES

Secrets Management — Never in Code

SECRETS IN CODE ARE PERMANENTLY COMPROMISED

Every secret committed to a git repository — even briefly, even deleted in the next commit — should be considered permanently compromised. Git history is immutable. Deleted files still exist in history. Forks and clones retain the history. Rotate immediately and investigate whether the secret was used.

```
# GitLeaks — scan current repo and all git history:
gitleaks detect --source=. --report-format json --report-path gitleaks.json
gitleaks detect --source=. --log-opts '--all'  # Scan ALL git history

# TruffleHog — find verified live secrets:
trufflehog git file://. --only-verified
trufflehog github --org=yourorg --only-verified

# Pre-commit hook — prevent secrets reaching repository:
# Install: pip install pre-commit && pre-commit install
# .pre-commit-config.yaml:
repos:
  - repo: https://github.com/gitleaks/gitleaks
    rev: v8.18.0
    hooks:
      - id: gitleaks
  - repo: https://github.com/Yelp/detect-secrets
    rev: v1.4.0
    hooks:
      - id: detect-secrets
        args: ['--baseline', '.secrets.baseline']

# If secret found in history — rotate the secret FIRST, then:
```

```
git filter-repo --invert-paths --path-glob '**/.env'  # Remove file from history
# Or use BFG Repo Cleaner (faster for large repos):
bfg --delete-files .env my-repo.git
```

Secrets Management Platforms

Platform	Best For	Key Features	Cost
HashiCorp Vault	All environments — self-hosted or cloud	Dynamic secrets, lease management, audit logs, fine-grained policies	Open source (self-host) or HCP Vault (managed)
AWS Secrets Manager	AWS-native workloads	Automatic rotation, CloudTrail audit, KMS encryption, native integrations	$0.40/secret/month + $0.05/10,000 API calls
Azure Key Vault	Azure-native workloads	RBAC, soft delete, purge protection, HSM-backed	Standard: $0.03/operation; Premium: $1/key/month (HSM)
GCP Secret Manager	GCP-native workloads	Automatic replication, version management, CMEK, audit logs	$0.06/secret/month + $0.03/10,000 access operations
Doppler	Cross-cloud, developer-friendly	Sync secrets to all cloud platforms, development workflow integration	Free for individuals; Team from $6/user/month

CHAPTER 08

CI/CD Security Pipeline

A complete security gate at every stage of the pipeline — automated, fast, and blocking.

GITHUB ACTIONS | SECURITY GATES | BLOCKING | AUTOMATED

Complete Security Pipeline — GitHub Actions

```
name: HorizonShield Security Pipeline
on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

jobs:
  security-scan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0  # Full history for
secret scanning

      # STAGE 1: Secrets detection — block
immediately if found
      - name: GitLeaks — Secrets Scan
        uses: gitleaks/gitleaks-action@v2
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN
}}
          GITLEAKS_ENABLE_COMMENTS: true  #
Comment on PR with findings

      # STAGE 2: SAST — static code analysis
      - name: Semgrep — SAST
        uses: returntocorp/semgrep-action@v1
        with:
```

```
          config: >
            p/owasp-top-ten
            p/cwe-top-25
            p/secrets
          publishDeployment: true
          publishToken: ${{
secrets.SEMGREP_APP_TOKEN }}

      # STAGE 3: SCA — dependency vulnerabilities
      - name: Dependency Review — CRITICAL/HIGH
only
        uses: actions/dependency-review-action@v4
        with:
          fail-on-severity: critical
          allow-licenses: MIT, Apache-2.0, BSD-2-
Clause

      # STAGE 4: Container image scanning
      - name: Build container image
        run: docker build -t app:${{ github.sha
}} .

      - name: Trivy — Container scan
        uses: aquasecurity/trivy-action@master
        with:
          image-ref: app:${{ github.sha }}
          format: sarif
          output: trivy-results.sarif
          severity: CRITICAL,HIGH
          exit-code: '1'  # Block pipeline on
CRITICAL/HIGH

      # STAGE 5: IaC scanning
      - name: Checkov — Terraform/CloudFormation
scan
        uses: bridgecrewio/checkov-action@master
        with:
          directory: ./infrastructure
          framework: terraform
          soft_fail: false
          output_format: sarif

      # Upload all findings to GitHub Security
tab
      - name: Upload SARIF results
        uses: github/codeql-action/upload-
sarif@v3
        if: always()
        with:
          sarif_file: '.'
```

HS

H O R I Z O N S H I E L D

Securing Your Digital Future, Today.

DevSecOps Implementation Guide 2026

Part of the HorizonShield Security Series — 16 comprehensive professional cybersecurity manuals covering every domain of modern enterprise security.

horizonshield.net · support@horizonshield.net