Software Development and Management Tools

Implementation with GitHub Actions

Radu Darius, Roman Dariana

Phase 4: Security Implementation

December 5, 2025

Project Overview

Project Goal:

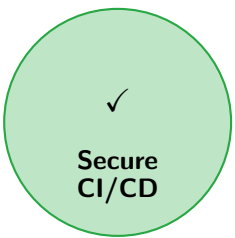
- Implement security controls for CI/CD pipeline
- Platform: GitHub Actions
- Environment: Automated security enforcement

Phase 4 Focus:

- Access Control
- Secret Management

Methodology:

- **Threat Model:** STRIDE
- **Approach:** Defense-in-depth
- **Validation:** Attack simulations



Threat Landscape: STRIDE Model

Identity & Access

- × **Spoofing:** Stolen tokens impersonate developers
- × **Repudiation:** Unsigned commits, no attribution

Data & Integrity

- × **Tampering:** Modified repos, CI configs
- × **Info Disclosure:** Secrets leaked in logs

System Abuse

- ! **DoS:** Pipeline abuse (build storms)
- ! **Elevation:** CI escalates to production

HIGH RISK

Addressed in Phase 4.1 & 4.2

MEDIUM RISK

Planned for Phase 4.3+

Access Control: Preventing Unauthorized Changes

✓ **Implemented & Tested**

1. Branch Protection

- Requires PR + reviews
- Status checks must pass
- No direct pushes

Mitigates: Tampering

2. Signed Commits

- Cryptographically signed (SSH/GPG)
- Verified via GitHub API
- Unsigned commits blocked

Mitigates: Repudiation, Spoofing

3. CODEOWNERS

- Critical files need approval
- Workflows, secrets, auth
- Automatic review requests

Mitigates: Tampering

3-Layer Defense: Branch Protection + Signed Commits + CODEOWNERS

Secret Management: Protecting Sensitive Data

✓ **Implemented & Tested**

1. GitHub Secret Scanning

- Detects 200+ secret types
- Push protection blocks commits
- Real-time monitoring

2. Encrypted Storage

- Secrets stored encrypted
- Masked in logs (shown as ***)
- Not accessible to forks

3. Leak Detection Workflow

- Custom patterns (passwords, keys, tokens)
- Scans all PRs and commits
- Blocks merge if secrets found

4. .gitignore Protection

- Prevents .env, credentials commits
- Blocks sensitive file types
- Source-level prevention

4-Layer Defense: Scan + Encrypt + Mask + Block

All layers mitigate: Information Disclosure

Security Controls Mapping

STRIDE Threat	Security Control	Implementation	Status
Spoofing	Authentication	MFA + SSH signatures	✓
Tampering	Access Control	Branch protection + CODEOWNERS	✓
Repudiation	Commit Signing	Cryptographic signatures (CI)	✓
Info Disclosure	Secret Management	Scanning + masking + detection	✓
DoS	Pipeline Hardening	Docker limits, SAST	··· Planned
Elevation	Isolation	Container isolation, scanning	··· Planned

4/6 Threats Fully Addressed

2/6 in Progress (Phase 4.3+)

Testing & Validation Results

Security Validation: Tested Attack Scenarios

✓ **Unsigned Commit**
- Attempted push without signature
- **Result:** CI blocked, PR failed
- **Control:** Signature enforcement

✓ **Secret Leak**
- Committed `password = "test123"`
- **Result:** Workflow detected & blocked
- **Control:** Secret leak detection

✓ **Unauthorized Edit**
- Modified CI workflow
- **Result:** Review request generated
- **Control:** CODEOWNERS

✓ **Direct Push**
- Bypassed PR process
- **Result:** Push rejected
- **Control:** Branch protection

✓ **Secret in Logs**
- Used secrets in workflow
- **Result:** Values masked as ***
- **Control:** Automatic masking

Security Metrics:
- 100% commit signing
- 100% MFA enabled
- 0 leaked secrets

Impact & Next Steps

Security Impact Achieved:

- ✓ **Tampering Prevention**
  Branch protection + signed commits + CODEOWNERS
- ✓ **Secret Protection**
  Scanning + encryption + masking + leak detection
- ✓ **Attribution**
  100% signed commits with cryptographic proof
- ✓ **Automation**
  CI/CD validates all changes

Next Steps (Phase 4.3 & 4.4):

- ··· **Pipeline Hardening:**
  - Docker containerization
  - SAST (static analysis)
  - Resource limits
- ··· **Supply Chain Security:**
  - Container image scanning
  - Dependency checks
  - Artifact signing
  - SBOM generation

Key Takeaway: Defense-in-depth with multiple overlapping controls ensures robust security

Thank You

Questions?