/secure-workflow-guide

Source: `~/.claude/skills/tob-building-secure-contracts/skills/secure-workflow-guide/SKILL.md`

---

name: secure-workflow-guide
description: Guides through Trail of Bits' 5-step secure development workflow. Runs Slither scans, checks special features (upgradeability/ERC conformance/token integration), generates visual security diagrams, helps document security properties for fuzzing/verification, and reviews manual security areas.

Secure Workflow Guide

Purpose

Guides through Trail of Bits' secure development workflow - a 5-step process to enhance smart contract security throughout development.

Use this: On every check-in, before deployment, or when you want a security review

---

The 5-Step Workflow

Covers a security workflow including:

Step 1: Check for Known Security Issues

Run Slither with 70+ built-in detectors to find common vulnerabilities:

- Parse findings by severity
- Explain each issue with file references
- Recommend fixes
- Help triage false positives

Goal: Clean Slither report or documented triages

Step 2: Check Special Features

Detect and validate applicable features:

- **Upgradeability**: slither-check-upgradeability (17 upgrade risks)
- **ERC conformance**: slither-check-erc (6 common specs)
- **Token integration**: Recommend token-integration-analyzer skill
- **Security properties**: slither-prop for ERC20

Note: Only runs checks that apply to your codebase

Step 3: Visual Security Inspection

Generate 3 security diagrams:

- **Inheritance graph**: Identify shadowing and C3 linearization issues
- **Function summary**: Show visibility and access controls
- **Variables and authorization**: Map who can write to state variables

Review each diagram for security concerns

Step 4: Document Security Properties

Help document critical security properties:

- State machine transitions and invariants
- Access control requirements
- Arithmetic constraints and precision
- External interaction safety
- Standards conformance

Then set up testing:

- **Echidna**: Property-based fuzzing with invariants
- **Manticore**: Formal verification with symbolic execution
- **Custom Slither checks**: Project-specific business logic

Note: Most important activity for security

Step 5: Manual Review Areas

Analyze areas automated tools miss:

- **Privacy**: On-chain secrets, commit-reveal needs
- **Front-running**: Slippage protection, ordering risks, MEV
- **Cryptography**: Weak randomness, signature issues, hash collisions
- **DeFi interactions**: Oracle manipulation, flash loans, protocol assumptions

Search codebase for these patterns and flag risks

For detailed instructions, commands, and explanations for each step, see WORKFLOW_STEPS.md.

How I Work

When invoked, I will:

1. **Explore your codebase** to understand structure
2. **Run Step 1**: Slither security scan
3. **Detect and run Step 2**: Special feature checks (only what applies)
4. **Generate Step 3**: Visual security diagrams
5. **Guide Step 4**: Security property documentation

6. **Analyze Step 5**: Manual review areas
7. **Provide action plan**: Prioritized fixes and next steps

Adapts based on:

- What tools you have installed
- What's applicable to your project
- Where you are in development

Rationalizations (Do Not Skip)

Rationalization	Why It's Wrong	Required Action
"Slither not available, I'll check manually"	Manual checking misses 70+ detector patterns	Install and run Slither, or document why it's blocked
"Can't generate diagrams, I'll describe the architecture"	Descriptions aren't visual - diagrams reveal patterns text misses	Execute slither --print commands, generate actual visual outputs
"No upgrades detected, skip upgradeability checks"	Proxies and upgrades are often implicit or planned	Verify with codebase search before skipping Step 2 checks
"Not a token, skip ERC checks"	Tokens can be integrated without obvious ERC inheritance	Check for token interactions, transfers, balances before skipping
"Can't set up Echidna now, suggesting it for later"	Property-based testing is Step 4, not optional	Document properties now, set up fuzzing infrastructure
"No DeFi interactions, skip oracle/flash loan checks"	DeFi patterns appear in unexpected places (price feeds, external calls)	Complete Step 5 manual review, search codebase for patterns
"This step doesn't apply to my project"	"Not applicable" without verification = missed vulnerabilities	Verify with explicit codebase search before declaring N/A
"I'll provide generic security advice instead of running workflow"	Generic advice isn't actionable, workflow finds specific issues	Execute all 5 steps, generate project-specific findings with file:line references

---

Example Output

When I complete the workflow, you'll get a comprehensive security report covering:

- **Step 1**: Slither findings with severity, file references, and fix recommendations
- **Step 2**: Special feature validation results (upgradeability, ERC conformance, etc.)
- **Step 3**: Visual diagrams analyzing inheritance, functions, and state variable authorization
- **Step 4**: Documented security properties and testing setup (Echidna/Manticore)
- **Step 5**: Manual review findings (privacy, front-running, cryptography, DeFi risks)

- **Action plan**: Critical/high/medium priority tasks with effort estimates
- **Workflow checklist**: Progress on all 5 steps

For a complete example workflow report, see EXAMPLE_REPORT.md.

What You'll Get

Security Report:

- Slither findings with severity and fixes
- Special feature validation results
- Visual diagrams (PNG/PDF)
- Manual review findings

Action Plan:

- ☐ Critical issues to fix immediately
- ☐ Security properties to document
- ☐ Testing to set up (Echidna/Manticore)
- ☐ Manual areas to review

Workflow Checklist:

- ☐ Clean Slither report
- ☐ Special features validated
- ☐ Visual inspection complete
- ☐ Properties documented
- ☐ Manual review done

Getting Help

Trail of Bits Resources:

- Office Hours: Every Tuesday (schedule)
- Empire Hacking Slack: #crytic and #ethereum channels

Other Security:

- Remember: Security is about more than smart contracts
- Off-chain security (owner keys, infrastructure) equally critical

---

Ready to Start

Let me know when you're ready and I'll run through the workflow with your codebase!

Revision #4
Created 2026-02-18 08:40:04 UTC by John
Updated 2026-05-31 20:01:41 UTC by John