/fix-review

Source: `~/.claude/skills/tob-fix-review/skills/fix-review/SKILL.md`

---

name: fix-review description: > Verifies that git commits address security audit findings without introducing bugs. This skill should be used when the user asks to "verify these commits fix the audit findings", "check if TOB-XXX was addressed", "review the fix branch", "validate remediation commits", "did these changes address the security report", "post-audit remediation review", "compare fix commits to audit report", or when reviewing commits against security audit reports. allowed-tools:

- Read
- Write
- Grep
- Glob
- Bash
- WebFetch

Fix Review

Differential analysis to verify commits address security findings without introducing bugs.

When to Use

- Reviewing fix branches against security audit reports
- Validating that remediation commits actually address findings
- Checking if specific findings (TOB-XXX format) have been fixed
- Analyzing commit ranges for bug introduction patterns
- Cross-referencing code changes with audit recommendations

When NOT to Use

- Initial security audits (use audit-context-building or differential-review)
- Code review without a specific baseline or finding set
- Greenfield development with no prior audit
- Documentation-only changes

Rationalizations (Do Not Skip)

Rationalization	Why It's Wrong	Required Action
"The commit message says it fixes TOB-XXX"	Messages lie; code tells truth	Verify the actual code change addresses the finding
"Small fix, no new bugs possible"	Small changes cause big bugs	Analyze all changes for anti-patterns
"I'll check the important findings"	All findings matter	Systematically check every finding
"The tests pass"	Tests may not cover the fix	Verify fix logic, not just test status
"Same developer, they know the code"	Familiarity breeds blind spots	Fresh analysis of every change

---

Quick Reference

Input Requirements

Input	Required	Format
Source commit	Yes	Git commit hash or ref (baseline before fixes)
Target commit(s)	Yes	One or more commit hashes to analyze
Security report	No	Local path, URL, or Google Drive link

Finding Status Values

Status	Meaning
FIXED	Code change directly addresses the finding
PARTIALLY_FIXED	Some aspects addressed, others remain
NOT_ADDRESSED	No relevant changes found
CANNOT_DETERMINE	Insufficient context to verify

Workflow

Phase 1: Input Gathering

Collect required inputs from user:

```
Source commit:  [hash/ref before fixes]
Target commit:  [hash/ref to analyze]
Report:         [optional: path, URL, or "none"]
```

If user provides multiple target commits, process each separately with the same source.

Phase 2: Report Retrieval

When a security report is provided, retrieve it based on format:

Local file (PDF, MD, JSON, HTML): Read the file directly using the Read tool. Claude processes PDFs natively.

URL: Fetch web content using the WebFetch tool.

Google Drive URL that fails: See `references/report-parsing.md` for Google Drive fallback logic using `gdrive` CLI.

Phase 3: Finding Extraction

Parse the report to extract findings:

Trail of Bits format:

- Look for "Detailed Findings" section
- Extract findings matching pattern: `TOB-[A-Z]+-[0-9]+`
- Capture: ID, title, severity, description, affected files

Other formats:

- Numbered findings (Finding 1, Finding 2)
- Severity-based sections (Critical, High, Medium, Low)
- JSON with `findings` array

See `references/report-parsing.md` for detailed parsing strategies.

Phase 4: Commit Analysis

For each target commit, analyze the commit range:

```
# Get commit list from source to target
git log <source>..<target> --oneline

# Get full diff
git diff <source>..<target>

# Get changed files
git diff <source>..<target> --name-only
```

For each commit in the range:

1. Examine the diff for bug introduction patterns
2. Check for security anti-patterns (see `references/bug-detection.md`)
3. Map changes to relevant findings

Phase 5: Finding Verification

For each finding in the report:

1. **Identify relevant commits** - Match by:
   - File paths mentioned in finding
   - Function/variable names in finding description
   - Commit messages referencing the finding ID
2. **Verify the fix** - Check that:
   - The root cause is addressed (not just symptoms)
   - The fix follows the report's recommendation
   - No new vulnerabilities are introduced
3. **Assign status** - Based on evidence:
   - FIXED: Clear code change addresses the finding
   - PARTIALLY_FIXED: Some aspects fixed, others remain
   - NOT_ADDRESSED: No relevant changes
   - CANNOT_DETERMINE: Need more context
4. **Document evidence** - For each finding:
   - Commit hash(es) that address it
   - Specific file and line changes
   - How the fix addresses the root cause

See `references/finding-matching.md` for detailed matching strategies.

Phase 6: Output Generation

Generate two outputs:

1. Report file (`FIX_REVIEW_REPORT.md`):

```
# Fix Review Report

**Source:** <commit>
**Target:** <commit>
**Report:** <path or "none">
**Date:** <date>

## Executive Summary

[Brief overview: X findings reviewed, Y fixed, Z concerns]

## Finding Status

| ID | Title | Severity | Status | Evidence |
|----|-------|----------|--------|----------|
| TOB-XXX-1 | Finding title | High | FIXED | abc123 |
| TOB-XXX-2 | Another finding | Medium | NOT_ADDRESSED | - |

## Bug Introduction Concerns

[Any potential bugs or regressions detected in the changes]

## Per-Commit Analysis

### Commit abc123: "Fix reentrancy in withdraw()"

**Files changed:** contracts/Vault.sol
**Findings addressed:** TOB-XXX-1
**Concerns:** None

[Detailed analysis]

## Recommendations
```

```
[Any follow-up actions needed]
```

2. Conversation summary:

Provide a concise summary in the conversation:

- Total findings: X
- Fixed: Y
- Not addressed: Z
- Concerns: [list any bug introduction risks]

Bug Detection

Analyze commits for security anti-patterns. Key patterns to watch:

- Access control weakening (modifiers removed)
- Validation removal (require/assert deleted)
- Error handling reduction (try/catch removed)
- External call reordering (state after call)
- Integer operation changes (SafeMath removed)
- Cryptographic weakening

See `references/bug-detection.md` for comprehensive detection patterns and examples.

Integration with Other Skills

differential-review: For initial security review of changes (before audit)

issue-writer: To format findings into formal audit reports

audit-context-building: For deep context when analyzing complex fixes

---

Tips for Effective Reviews

Do:

- Verify the actual code change, not just commit messages
- Check that fixes address root causes, not symptoms
- Look for unintended side effects in adjacent code

- Cross-reference multiple findings that may interact
- Document evidence for every status assignment

Don't:

- Trust commit messages as proof of fix
- Skip findings because they seem minor
- Assume passing tests mean correct fixes
- Ignore changes outside the "fix" scope
- Mark FIXED without clear evidence

Reference Files

For detailed guidance, consult:

- `references/finding-matching.md` - Strategies for matching commits to findings
- `references/bug-detection.md` - Comprehensive anti-pattern detection
- `references/report-parsing.md` - Parsing different report formats, Google Drive fallback

---

Revision #5
Created 2026-02-18 08:40:07 UTC by John
Updated 2026-06-21 20:01:12 UTC by John