/differential-review

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

---

name: differential-review description: > Performs security-focused differential review of code changes (PRs, commits, diffs). Adapts analysis depth to codebase size, uses git history for context, calculates blast radius, checks test coverage, and generates comprehensive markdown reports. Automatically detects and prevents security regressions. allowed-tools:

- Read
- Write
- Grep
- Glob
- Bash

Differential Security Review

Security-focused code review for PRs, commits, and diffs.

Core Principles

1. **Risk-First**: Focus on auth, crypto, value transfer, external calls
2. **Evidence-Based**: Every finding backed by git history, line numbers, attack scenarios
3. **Adaptive**: Scale to codebase size (SMALL/MEDIUM/LARGE)
4. **Honest**: Explicitly state coverage limits and confidence level
5. **Output-Driven**: Always generate comprehensive markdown report file

---

Rationalizations (Do Not Skip)

Rationalization	Why It's Wrong	Required Action
"Small PR, quick review"	Heartbleed was 2 lines	Classify by RISK, not size
"I know this codebase"	Familiarity breeds blind spots	Build explicit baseline context
"Git history takes too long"	History reveals regressions	Never skip Phase 1
"Blast radius is obvious"	You'll miss transitive callers	Calculate quantitatively
"No tests = not my problem"	Missing tests = elevated risk rating	Flag in report, elevate severity
"Just a refactor, no security impact"	Refactors break invariants	Analyze as HIGH until proven LOW
"I'll explain verbally"	No artifact = findings lost	Always write report

---

Quick Reference

Codebase Size Strategy

Codebase Size	Strategy	Approach
SMALL (<20 files)	DEEP	Read all deps, full git blame
MEDIUM (20-200)	FOCUSED	1-hop deps, priority files
LARGE (200+)	SURGICAL	Critical paths only

Risk Level Triggers

Risk Level	Triggers
HIGH	Auth, crypto, external calls, value transfer, validation removal
MEDIUM	Business logic, state changes, new public APIs
LOW	Comments, tests, UI, logging

Workflow Overview

```
Pre-Analysis → Phase 0: Triage → Phase 1: Code Analysis → Phase 2: Test Coverage
    ↓              ↓                    ↓                        ↓
Phase 3: Blast Radius → Phase 4: Deep Context → Phase 5: Adversarial → Phase 6: Report
```

Decision Tree

Starting a review?

```
├─ Need detailed phase-by-phase methodology?
│  └─ Read: methodology.md
│     (Pre-Analysis + Phases 0-4: triage, code analysis, test coverage, blast radius)
│
├─ Analyzing HIGH RISK change?
│  └─ Read: adversarial.md
│     (Phase 5: Attacker modeling, exploit scenarios, exploitability rating)
│
├─ Writing the final report?
│  └─ Read: reporting.md
│     (Phase 6: Report structure, templates, formatting guidelines)
│
├─ Looking for specific vulnerability patterns?
│  └─ Read: patterns.md
│     (Regressions, reentrancy, access control, overflow, etc.)
│
└─ Quick triage only?
   └─ Use Quick Reference above, skip detailed docs
```

Quality Checklist

Before delivering:

- ☐ All changed files analyzed
- ☐ Git blame on removed security code
- ☐ Blast radius calculated for HIGH risk
- ☐ Attack scenarios are concrete (not generic)

- [ ] Findings reference specific line numbers + commits
- [ ] Report file generated
- [ ] User notified with summary

Integration

audit-context-building skill:

- Pre-Analysis: Build baseline context
- Phase 4: Deep context on HIGH RISK changes

issue-writer skill:

- Transform findings into formal audit reports
- Command: `issue-writer --input DIFFERENTIAL_REVIEW_REPORT.md --format audit-report`

Example Usage

Quick Triage (Small PR)

```
Input: 5 file PR, 2 HIGH RISK files
Strategy: Use Quick Reference
1. Classify risk level per file (2 HIGH, 3 LOW)
2. Focus on 2 HIGH files only
3. Git blame removed code
4. Generate minimal report
Time: ~30 minutes
```

Standard Review (Medium Codebase)

```
Input: 80 files, 12 HIGH RISK changes
Strategy: FOCUSED (see methodology.md)
1. Full workflow on HIGH RISK files
2. Surface scan on MEDIUM
3. Skip LOW risk files
```

```
4. Complete report with all sections
Time: ~3-4 hours
```

Deep Audit (Large, Critical Change)

```
Input: 450 files, auth system rewrite
Strategy: SURGICAL + audit-context-building
1. Baseline context with audit-context-building
2. Deep analysis on auth changes only
3. Blast radius analysis
4. Adversarial modeling
5. Comprehensive report
Time: ~6-8 hours
```

When NOT to Use This Skill

- **Greenfield code** (no baseline to compare)
- **Documentation-only changes** (no security impact)
- **Formatting/linting** (cosmetic changes)
- **User explicitly requests quick summary only** (they accept risk)

For these cases, use standard code review instead.

Red Flags (Stop and Investigate)

Immediate escalation triggers:

- Removed code from "security", "CVE", or "fix" commits
- Access control modifiers removed (onlyOwner, internal → external)
- Validation removed without replacement
- External calls added without checks
- High blast radius (50+ callers) + HIGH risk change

These patterns require adversarial analysis even in quick triage.

Tips for Best Results

Do:

- Start with git blame for removed code
- Calculate blast radius early to prioritize
- Generate concrete attack scenarios
- Reference specific line numbers and commits
- Be honest about coverage limitations
- Always generate the output file

Don't:

- Skip git history analysis
- Make generic findings without evidence
- Claim full analysis when time-limited
- Forget to check test coverage
- Miss high blast radius changes
- Output report only to chat (file required)

Supporting Documentation

- **methodology.md** - Detailed phase-by-phase workflow (Phases 0-4)
- **adversarial.md** - Attacker modeling and exploit scenarios (Phase 5)
- **reporting.md** - Report structure and formatting (Phase 6)
- **patterns.md** - Common vulnerability patterns reference

For first-time users: Start with methodology.md to understand the complete workflow.

For experienced users: Use this page's Quick Reference and Decision Tree to navigate directly to needed content.

Revision #5
Created 2026-02-18 08:40:06 UTC by John
Updated 2026-06-21 20:01:11 UTC by John