Cytix

Recurring issue review

Identify the root cause of your code vulnerabilities

What's in the pack

Part 1
How to run your review

Part 2
The worksheet

Part 3
Your diagnosis

Part 4
Worked example

Part 1

How to run your review

How to use this pack

Pick one issue that's turned up at least twice, whether it's the same weakness type, the same part of the application or the same fix being applied again. Two occurrences are enough to start, and with three or more you can trust the pattern a bit more. Issues caught in code review count too. You'll need the tickets, pull requests, review notes and test or scan results behind each one. Keep answers short, as a few words per box is plenty.

Keep answers comparable

Patterns only show up if we describe them in the same way each time. Here are our recommendations for fields which we label ‘See answer options’, but use terminology which best fits your organisation.

Description	Answer options
How it was found	Code review, automated test, scanner, pentest, bug bounty, production incident, other
Type of change	New endpoint or API, new feature, bug fix, refactor, dependency upgrade, config or infrastructure, data or schema migration, permission or role change, other
Main security area	Authentication, authorisation and permissions, sensitive data, external input, secrets and keys, third-party integration, none flagged
Review it got	Peer code review, security review, threat model, no review
Scope of the fix	Local to this change, shared component or library, process or checklist, not fixed yet

Spotting the common thread

Work across each row of the worksheet. When the same answer appears in more than one column, tick the ‘Same?’ box. Those ticks are your cue to look closer.

A tick isn't proof on its own, though. If nearly all your changes go through peer review, a repeated 'peer code review' tells you less than a repeated service, type of change or security area. Ask whether the match is unusual for your team.

Choosing a solution

Pick the lightest change that would have caught every occurrence, not just the latest one. Good options include a shared library or middleware, a secure default in your framework, a question on the review checklist, a test that runs on every change of that kind, or clearer code ownership. Agree who'll put it in place.

After the review

Add one or two questions to your PR template so future records are easier to compare, such as 'What does this change touch from a security point of view?' and 'How risky does it look, and why?'. Link every security fix back to the change it came from, and tag the weakness type with its CWE category. Then book the same 30 minutes for next quarter. If the issue hasn't reappeared on new changes of that kind, your guardrail is working.

Cytix

Keeping change records complete gets harder as you grow. Cytix keeps a connected audit trail for every change it assesses, so the record is already there when you need it. Find out more at cytix.io

Part 2

The worksheet

One column per occurrence. Tick ‘Same?’ wherever an answer matches across columns

Issue we’re reviewing

Field	Occurrence 1	Occurrence 2	Occurrence 3	Occurrence 4	Same?
Step 1. The issue					
Finding or issue					
Weakness type (CWE) e.g. CWE-639					
Date found					
How it was found See answer options					
Step 2. What changed					
Change that introduced it Ticket, PR or release					
Service or component					
Team					
Type of change See answer options					
How risky did it look at the time Low, medium, high or not assessed					
Step 3. What it touched					
Main security area See answer options					
Was that area flagged at the time? Yes, no or not sure					
Step 4. The decisions					
Review it got See answer options					
Was a risk accepted? Yes, no or not sure					

Field	Occurrence 1	Occurrence 2	Occurrence 3	Occurrence 4	Same?
Step 5. The fix					
How it was fixed					
Scope of the fix See answers options					
Was the fix retested? Yes, no or not sure					

Part 3

Your diagnosis

Which rows had ticks?

What do the records have in common?

What will we change so it doesn’t come back?

Part 4

Worked example (illustrative)

A finished worksheet for a made-up team. Matching answers are shaded and ticked so you can see where the common thread sits

Issue we're reviewing
Broken object level authorisation, where users can reach other users' data by changing an ID in a request

Field	Occurrence 1	Occurrence 2	Occurrence 3	Occurrence 4	Same?
Step 1. The issue					
Finding or issue	Order export returns other customer's orders	Invoice download accepts any invoice ID	Profile API returns any user's profile		
Weakness type (CWE) e.g. CWE-639	CWE-639	CWE-639	CWE-639		✓
Date found	March	May	June		
How it was found See answer options	Pentest	Automated test	Pentest		✓
Step 2. What changed					
Change that introduced it Ticket, PR or release	PR adding the order export document	PR adding invoice downloads	PR adding the profile API		
Service or component	Orders service	Billing service	Accounts service		
Team	Checkout team	Payments team	Identity team		
Type of change See answer options	New endpoint or API	New endpoint or API	New endpoint or API		✓
How risky did it look at the time Low, medium, high or not assessed	Low	Low	Medium		✓
Step 3. What it touched					
Main security area See answer options	Sensitive data	Sensitive data	Sensitive data		✓
Was that area flagged at the time? Yes, no or not sure	Yes	Yes	Yes		✓
Step 4. The decisions					
Review it got See answer options	Peer review	Peer review	Peer review		✓
Was a risk accepted? Yes, no or not sure	No	No	No		✓

Field	Occurrence 1	Occurrence 2	Occurrence 3	Occurrence 4	Same?
Step 5. The fix					
How it was fixed	Ownership check added in the export handler	Ownership check added in the download handler	Ownership check added in the profile handler		
Scope of the fix See answers options	Local to this change	Local to this change	Local to this change		✓
Was the fix retested? Yes, no or not sure	Yes	Yes	Yes		✓

Part 4

Worked example: the diagnosis

Which rows had ticks?

Weakness type, type of change, main security area, review it got and scope of the fix stood out. Some others matched too, like 'retested: yes', but that's true of almost every fix, so we set them aside.

What do the records have in common?

All three were new API endpoints returning customer data, built by different teams in different services. Each passed normal code review, and each fix was local to one handler.
There's no shared authorisation check for new routes, so every new endpoint starts without one.

What will we change so it doesn’t come back?

Add a shared authorisation layer that new routes use by default, and a review checklist question asking which check protects any new route. We'll check again at next quarter's review.

Cytix

Keeping change records complete gets harder as you grow. Cytix keeps a connected audit trail for every change it assesses, so the record is already there when you need it. Find out more at cytix.io.