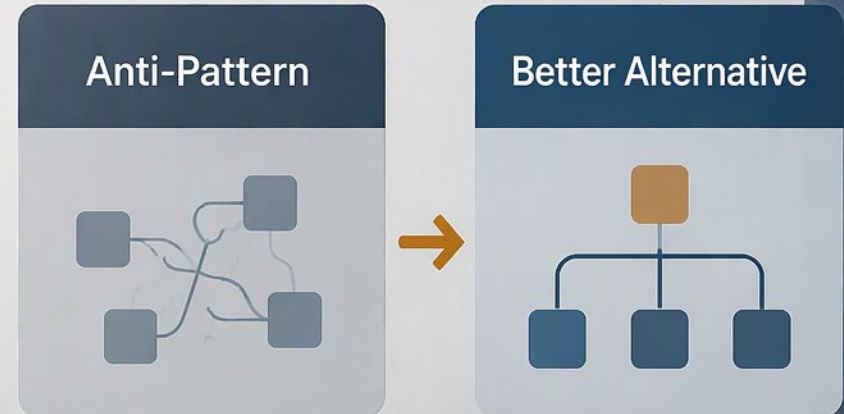


Cybersecurity Anti-Patterns and Better Alternatives

- Well-intentioned teams still ship weak designs: speed, inherited assumptions, late security input
- Anti-pattern: a repeatable response that looks helpful but creates disproportionate risk
- Scope: defensive design, architecture, and governance mistakes — no offensive instruction
- Rhythm: recognize the pattern, understand the failure, adopt the alternative, agree trade-offs
- Shared vocabulary lets product, IT, and security review designs together



Recognize



Understand



Adopt



Agree

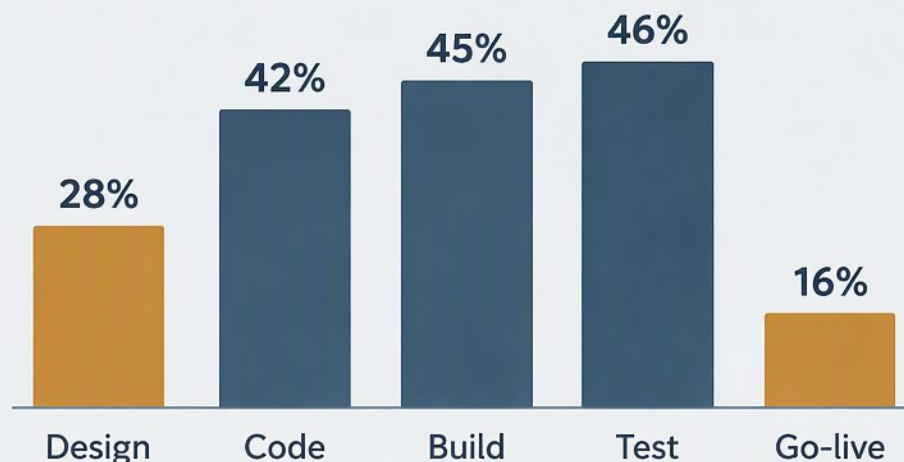
Why Defensive Weaknesses Persist

PEOPLE
PROCESSES
PERSPECTIVE
STRONGER OUTCOMES

- Shipping speed outranks security in most visible incentives
- Threat model debt: designs fixed early, never revisited
- Copy-paste architecture carries insecure defaults without ownership
- Security still gates late: controls cluster in code, build, test
- Siloed product, IT, and security teams each hold partial context
- 2025 survey: AppSec strongest in test (46%), build (45%), code (42%); weak at design (28%) and go-live (16%)
- Security debt now affects 82% of organizations

AppSec Strength by Development Phase

2025 survey results



Security debt now affects
82% of organizations

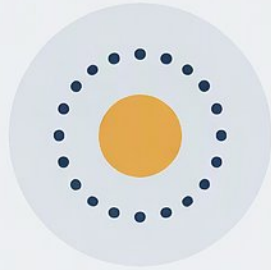
FROM A STRONGER
SECURITY BASELINE

TO A WEAKER
SECURITY STATE

Recognition Framework: **Four Lenses**



- **Trust assumptions:**
what does this design implicitly trust, and is that justified?



- **Exposure surface:**
what becomes reachable if one assumption fails?



- **Failure behavior:**
what happens when a check or credential is unavailable?



- **Governance ownership:**
who is accountable, and where is it recorded?



Signal questions work for technical and non-technical reviewers alike



Separate defensible, documented trade-offs from unrecognized anti-patterns

Anti-Patterns in Identity and Access



- Over-broad standing privileges and shared accounts that never get cleaned up
- Secrets embedded in code, config files, or deployment scripts instead of a vault
- Authentication and authorization decisions left in the client or UI layer
- Better alternatives: least privilege by default, centralized identity management, short-lived credentials
- Match authentication strength to data sensitivity — select assurance levels, don't assume them

Anti - Patterns in Data Handling and Privacy

Less Risk
Stronger Trust



Intake



Classification



Logging



Disposal

MINIMIZE RISK ACROSS THE DATA LIFECYCLE



Anti-Patterns

- Collect-now-classify-later: data hoarded because it may be useful, never minimized
- Copying production data into development and test environments
- Plain-text logging of tokens, session IDs, credentials, personal data, full request bodies



Better Alternatives

- Alternatives: classify and minimize at intake; mask, tokenize, or hash with reversibility limits in mind
- Apply retention and disposal to logs, backups, and debug output; protect logs centrally with restricted read access

DATA PEOPLE
HIGHER STANDARDS
BRIGHTER OUTCOMES

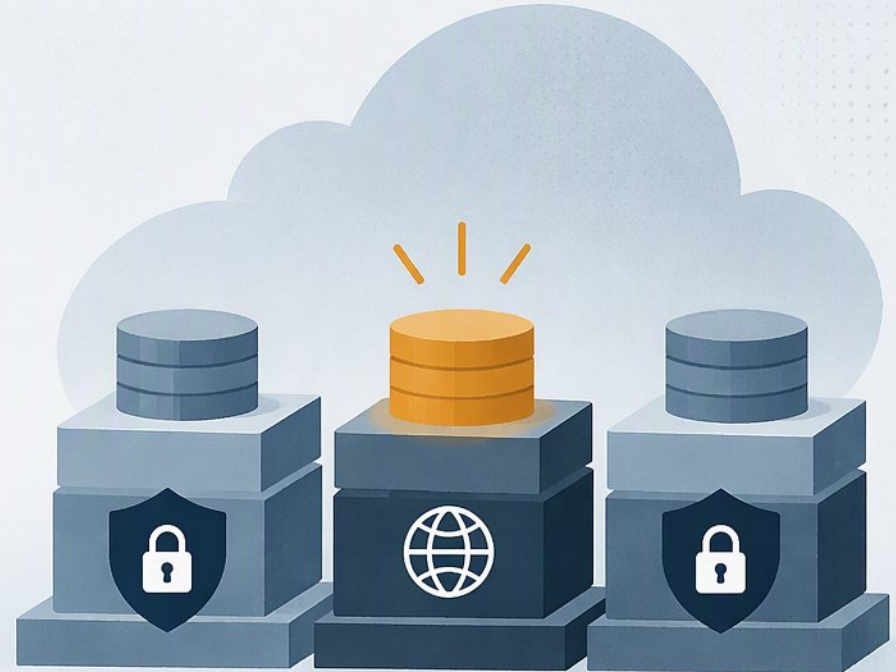
Anti-Patterns in Resilience and Failure Handling

- Failing open when a check, policy service, or dependency is unavailable
- Error messages, retries, and fallbacks treated as security-neutral choices
- Single points of trust and bypass paths that quietly become the real access route
- Better: fail-safe defaults, bounded retries, observable degradation, explicit trust decisions
- Design review question: during an outage, which control stops working — and what happens then?



Anti-Patterns in Cloud Configuration

- Default-open settings cause accidental public exposure of storage and admin interfaces
- Misconception: the provider secures everything — you always own data, identities, accounts, and access
- Green-field baselines drift to weaker states without detection or ownership
- Better: publish a secure baseline before design; enforce guardrails as policy-as-code
- Monitor for drift continuously; separate accounts or environments to limit blast radius



PEOPLE
PROCESSES
A MORE RESILIENT
TOMORROW

-

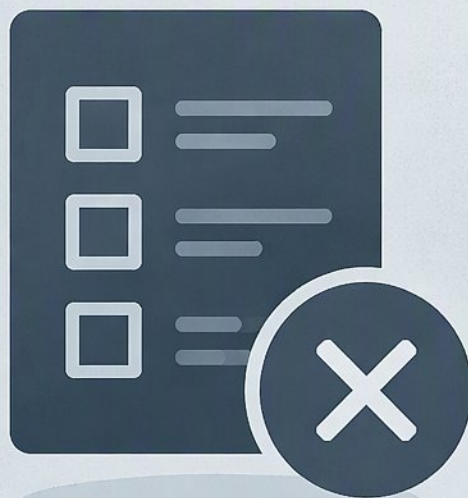


Anti-Patterns in Governance and Security Metrics

- Compliance checkbox exercises detached from the risk the product actually carries
- Vanity metrics — activity counts and coverage percentages — that hide exposure
- Unclear ownership: everyone responsible means no one accountable
- Better: outcome-based and risk-based metrics, paired KPIs and KRIs
- Explicit risk acceptance with a named owner, recorded where designs live, revisited on schedule or material change

Anti-Patterns

Activity without insight



Better Approach

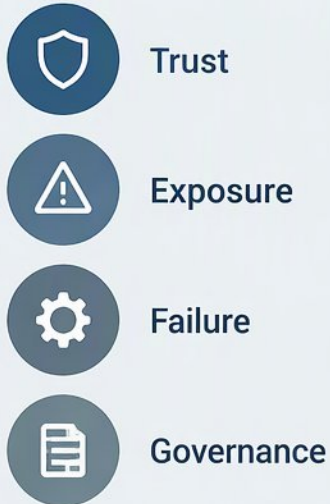
Outcomes, risk, and accountability



From Anti-Pattern to Alternative: A Repeatable Playbook

A practical approach to safer, more resilient solutions

Four-Lens Review



Workflow



Openers that raise concerns without triggering defensiveness across product, IT, and security



Small first steps for inherited anti-patterns versus planned remediation slots



Tie defensive work to business-critical processes, not abstract severity scores



Capture decisions in design docs, backlog, or risk register with named owners

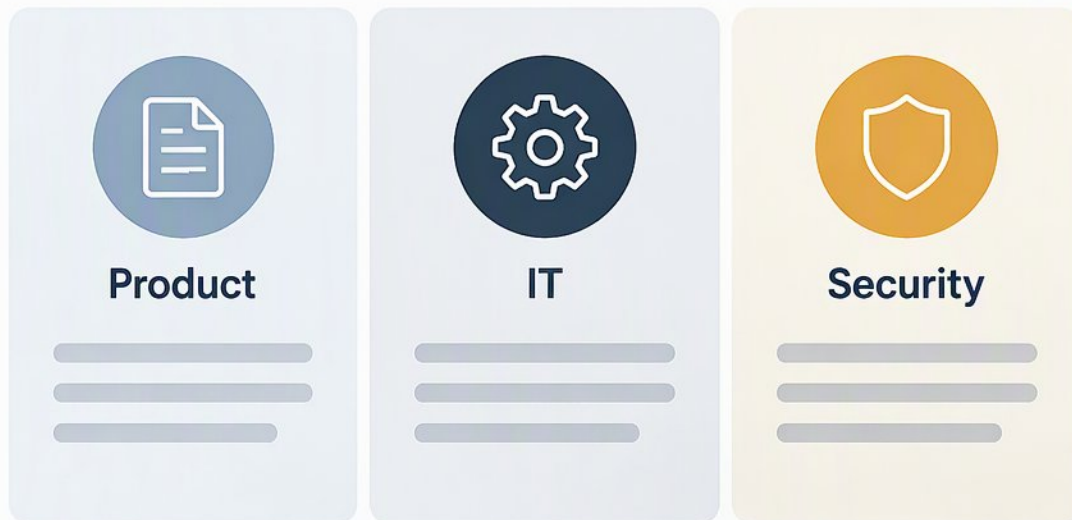


AppSec controls cluster in Test (46%), Build (45%), Code (42%); Design just 28% (Checkmarx 2025)

Workshop: **Spot the Anti-Pattern**

PEOPLE
PERSPECTIVE
BETTER DECISIONS

- Three short scenarios: product, IT, and security viewpoints — no offensive detail
- Work in small mixed-role groups to apply the four lenses
- Name the trust assumption that is actually failing
- Debrief: defensible trade-offs wrongly flagged, and anti-patterns waved through
- Capture the failing assumption and the decision, not just the symptom



Workshop: Choose the Alternative and Name the Trade-off

- Each group proposes a better alternative and states its cost
- Name what you trade: effort, latency, usability, or vendor dependency
- Write the decision so a reviewer could act on it
- Capture pattern, risk, alternative, owner, and review date
- Facilitator test: does it fix the failing trust assumption, or just add a nearby control?

BETTER
DECISIONS
STRONGER
OUTCOMES

Adds a nearby control



Addresses the symptom with an additional control, but the underlying trust assumption still fails.

OWNER:

REVIEW DATE:



Fixes the failing trust assumption



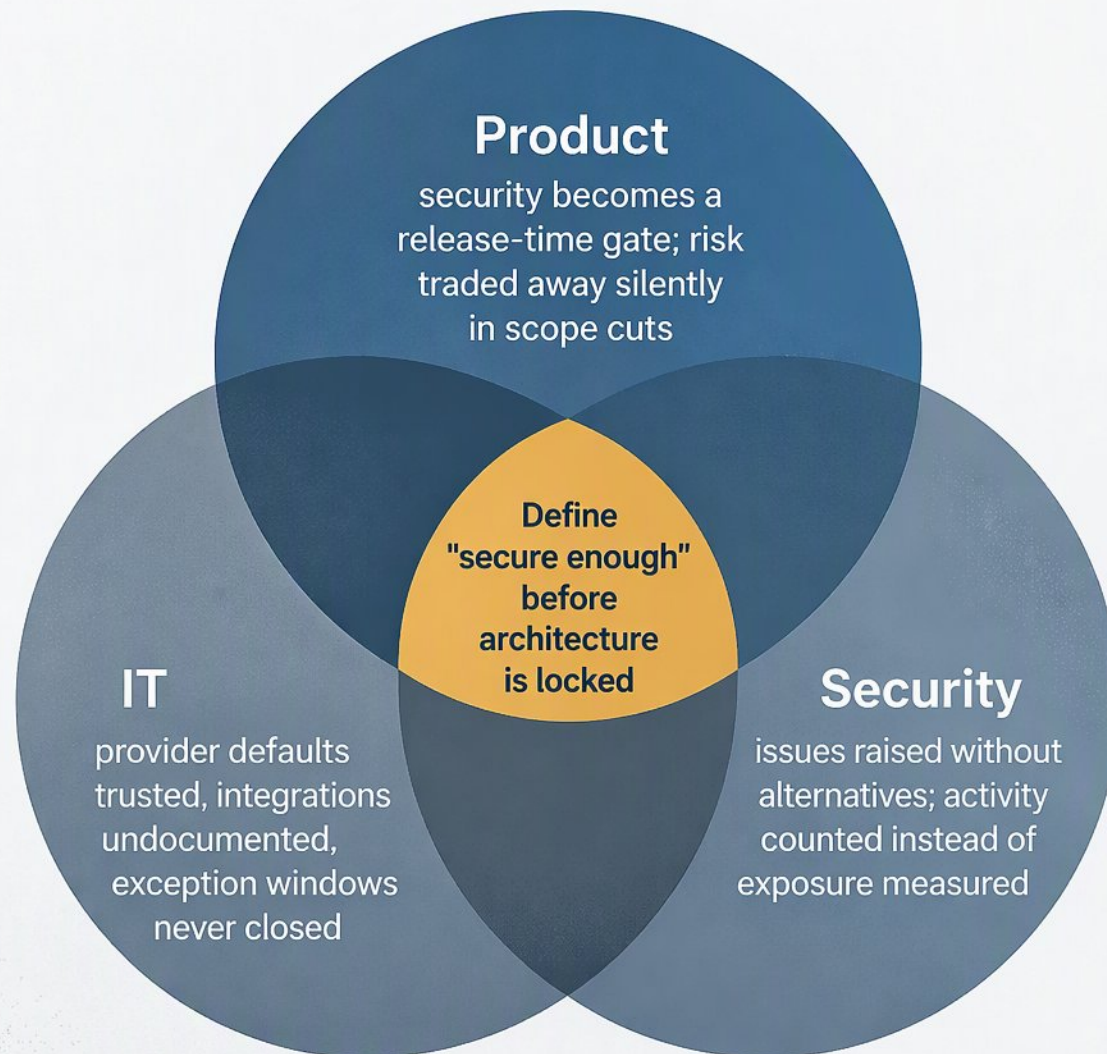
Removes or changes the failing trust assumption, addressing the root cause.

OWNER:

REVIEW DATE:

Role-Specific Blind Spots to Watch

PEOPLE
PERSPECTIVE
SAFER OUTCOMES



Security oversight is shifting to product teams — expect new accountability



Survey: only 20% report high DevSecOps maturity; 43% moved oversight to product teams

Four-Lens Review Frame



Trust



Exposure



Failure



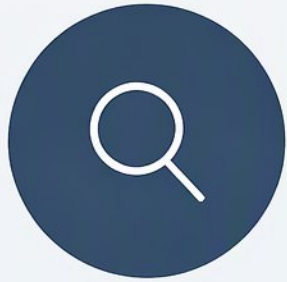
Governance

SHARED
INSIGHT
STRONGER
SYSTEMS

Commitments and Next Steps

FROM AWARENESS TO ACTION

SAFER
DESIGNS
STRONGER
OUTCOMES



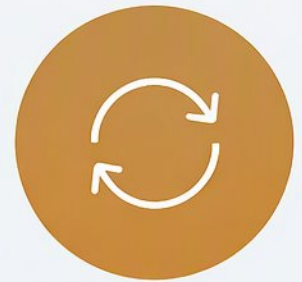
- Personal: pick one anti-pattern, name an owner, set a 30-day review date



- Team: make the four-lens review a standing design-review input



- Record decisions where designs live, not in a separate tracker



- Re-run the review when dependencies, data sensitivity, or infrastructure materially change

THE
FOUR-LENS
REVIEW



Trust



Exposure



Failure



Governance

PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/96236bb7/public