

1.1 THE QUESTION BEFORE THE CONTROLS

Security Begins with a System Boundary

Before you ask whether a system is secure, you have to decide which system your claim is actually about.

BY THE END OF THIS SEQUENCE, YOU SHOULD BE ABLE TO

- state a protected outcome
- choose a useful boundary
- name what could invalidate it

Security work often starts with a question that sounds simpler than it is: Is this system secure? The problem is hidden inside the noun. Which system? A payroll application can be drawn as a single rectangle on an architecture diagram, but payroll does not happen inside that rectangle. It depends on employee identities, HR records, privileged administrators, bank integrations, certificates, secrets, DNS, cloud infrastructure, endpoint devices, software updates, backups, monitoring, vendor support paths, and people who approve unusual transactions.

That distinction matters because a local control can be perfectly healthy while the outcome it is supposed to protect is already lost. Imagine that the payroll application correctly prevents ordinary users from changing bank-routing information. If a help-desk process can reset an administrator account, a deployment pipeline can ship code that bypasses authorization, or a database administrator can alter the record directly, the application check is not the whole security story.

A boundary is therefore not a clerical line around whatever team happens to own. It is an instrument of reasoning. Draw it too narrowly and inherited risk disappears. Draw it so broadly that everything is included and the model becomes useless. The practical target is the smallest boundary that preserves the protected outcome, expanded whenever something outside the line can materially make the claim false.

WORKED MINIATURE: ONE CLAIM, THREE POSSIBLE BOUNDARIES

CLAIM

Only authorized payroll administrators can change bank-routing information.

TOO NARROW

The payroll UI and its role check. Precise, but blind to identity recovery, deployment, and direct database authority.

USEFUL START

Payroll app + identity + deployment + datastore + privileged paths. Expand further only when evidence shows another causal dependency.

KEY INSIGHT

Do not start with the box. Start with the claim. The boundary earns its shape from what must remain true and what can make that truth fail.

1.2 PROTECTED OUTCOMES

Make “Secure” Testable

Security properties become useful when they are tied to an outcome somebody actually depends on.

Words such as confidentiality, integrity, availability, authenticity, and accountability are valuable abstractions. But a real system rarely exists to maximize an abstraction. It exists to produce an outcome. “Payroll is secure” is too vague to test; almost any control can be made to sound relevant to it.

A STRONGER PROTECTED-OUTCOME STATEMENT

Authorized employees receive the correct amount on time; unauthorized parties cannot alter payment instructions; sensitive tax and banking information is not improperly disclosed; and the organization can recover from material corruption before the next payroll deadline.

That sentence immediately gives the analyst more to work with. Integrity matters because payment instructions must be correct. Availability matters because payroll has a deadline. Confidentiality matters because bank and tax data are sensitive. Identity and authorization matter because only legitimate people and processes should change payroll state. Recoverability matters because prevention may fail.

It also exposes dependencies. Correct payroll depends on source employment data, identity, calculation logic, bank connectivity, approval workflows, and recoverable state. A security-relevant failure therefore does not require an attacker exploiting software. A broken integration, mistaken administrative change, corrupt upstream record, expired certificate, failed key rotation, or unavailable identity provider can all destroy trust in the outcome.

USE FOUR QUESTIONS TO BUILD THE CLAIM

What must remain true?

Name the protected result in operational terms, not a control catalogue.

For whom does it matter?

Identify the people, services, operators, customers, or public outcomes that depend on it.

What kinds of failure matter?

Disclosure, corruption, unauthorized action, delay, unavailability, lost provenance, or failed recovery can imply different boundaries.

What is the time horizon?

An hour of downtime, a week of corrupted records, and years of confidentiality exposure are different architecture problems.

SELF-EXPLANATION CHECK

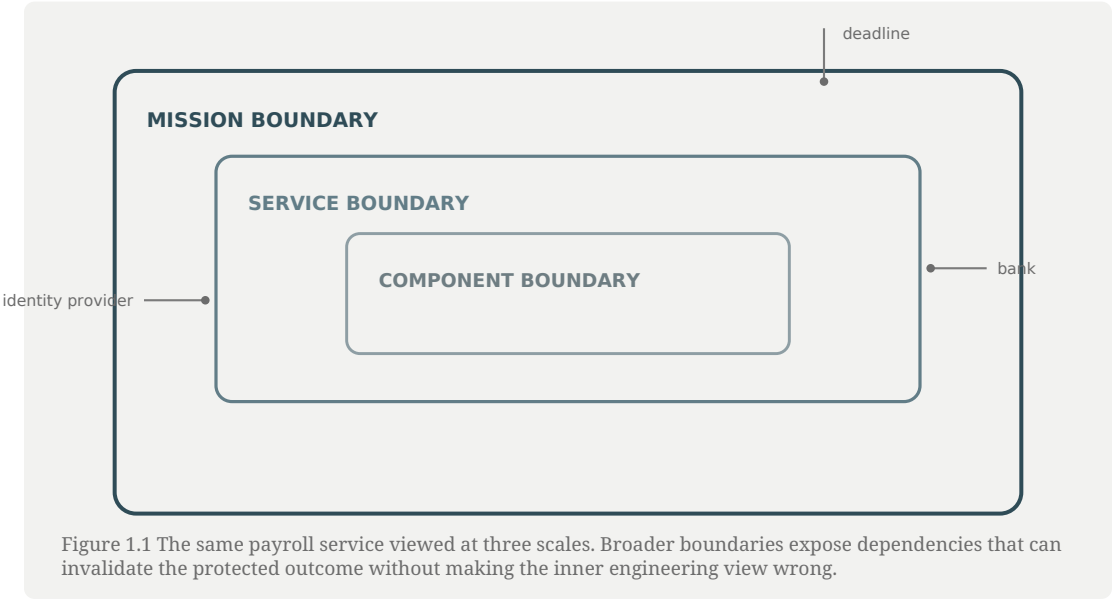
Suppose the payroll application is available and its role configuration is correct, but the identity provider is suspected compromised. Explain, in one or two sentences, why the protected outcome may still be unproven. Then name the first two dependencies you would add to the active boundary.

1.3 COGNITIVE ZOOM

Three Boundary Scales

Keep more than one view. Each scale answers a different claim, and each hides something the next scale can reveal.

A useful boundary is not one perfect diagram. It is an aperture you can change without losing the thread of the reasoning. For many systems, three nested views are enough to make that movement explicit: component, service, and mission.



1 COMPONENT

Local engineering surface: roles, workflow, datastore, and direct interfaces. Use it for precise control questions.

2 SERVICE

Adds identity, HR source, bank integration, administration, logging, recovery, deployment, and other dependencies required to deliver the service.

3 MISSION

Adds the business outcome and its constraints: payroll deadline, finance operations, employee impact, continuity, and legal obligations.

PROFESSIONAL HABIT

Before accepting a diagram, ask: What claim does this scale support? What can disappear when I zoom in? What outside dependency could make the conclusion false?

PRACTICE • BOUNDARY MODEL

Build It. Then Try to Break It.

The point is not a handsome diagram. The point is to expose where the conclusion gets its trust.

Use a lawful real system you are authorized to examine, or use the payroll service from these pages. Build one inspectable model that another person could challenge without needing you beside them to explain what you meant.

1 STATE THE PROTECTED OUTCOME

Write one sentence describing what must remain true, for whom, and on what time horizon.

3 ENUMERATE DEPENDENCIES

Add identities, services, data sources, control planes, suppliers, networks, keys, recovery systems, and people that cross the first line.

5 CLASSIFY WHAT YOU FIND

Bring material dependencies into the active model, or record them as explicit assumptions with invalidation conditions.

7 ADD RECOVERY DEPENDENCIES

Ask what must remain independently trustworthy to restore the service after a serious compromise.

2 NAME THE IMMEDIATE SYSTEM

List the smallest set of components directly responsible for producing that outcome.

4 ASK THE FALSIFICATION QUESTION

For each dependency: If this were compromised, unavailable, corrupted, or falsely trusted, could the protected claim become false?

6 MAP TRUST CROSSINGS

For each important crossing, state what is trusted, why, how much authority is granted, and how the relationship is observed or revoked.

8 CHANGE ONE CONDITION

Compromise identity, remove telemetry, fail a supplier, add an API, lose a region, or shorten the recovery deadline. Revise only what the changed condition materially affects.

BREAK IT DELIBERATELY

Assume the enterprise identity provider is suspected compromised. Your payroll application's local role configuration may still be correct, but the basis on which it accepts user identity is no longer trustworthy. Reopen the model: Which active sessions exist? Can privileged access be revoked independently? Do vendor identities federate through the same provider? Can the bank integration be changed by identities rooted in the compromised plane? Can you still trust the logs?

PASS EVIDENCE

Another practitioner can explain what the model claims, where its proof stops, which dependencies can invalidate it, and why the changed condition caused the boundary to move.

FAILURE EVIDENCE

The model is a generic enterprise picture; exclusions are unstated; ownership is treated as causal scope; trust crossings are unlabeled; or one changed condition forces a complete restart.

TRANSFER

Choose a different environment entirely: a hospital medication system, a source-code release pipeline, a home automation system, or another service you understand. Change the nouns. Preserve the reasoning.