WACK CASE FILE / WACK-CONTENT-0063 / SEASON 1 FILE 04

THE TEMPORARY FIREWALL RULE THAT BECAME ARCHITECTURE

The exception had an expiration date. The dependencies it attracted did not.

> **"Approved for fourteen days. Renewed for four years."**

FICTIONAL COMPOSITE / REAL PRACTITIONER PATTERN

Fictional composite synthesized from common practitioner patterns. No client, assessment, or incident is described. Practitioner education, not PCI SSC terminology or a compliance determination. Independent resource; not affiliated with or endorsed by PCI SSC.

COLD OPEN

The firewall rule was approved for fourteen days, renewed for four years, and eventually listed as a prerequisite in a system nobody had heard of when the exception began.

THE SCENE

The original request was reasonable. A migration had stalled, a legacy application needed a short bridge, and the team documented a narrow rule so work could continue. The exception record named a requester, business reason, and end date. At the time, the path connected two systems and everyone expected one of them to disappear before the next quarter.

The system did not disappear. The migration shifted, the owner changed roles, and the bridge stayed quiet enough to avoid attention. New teams discovered that the route worked and connected their own processes. A troubleshooting guide referenced it. An automation job depended on it. A template copied part of the configuration. The exception remained one line in a register while its Wack Radius expanded across the architecture.

NARRATIVE TURN / 01

THE ANNUAL RENEWAL RITUAL

Each year, the governance team asked whether the exception was still needed. The operational answer was yes because removing it might break something. The evidence for that answer was the fact that nothing had broken while it remained. Risk acceptance became a ritual: update the date, preserve the wording, route the approval, and promise to revisit the migration.

No one was being lazy. The problem was structural. The people approving the exception could see its original purpose but not its current consumers. The people who knew individual consumers did not know those dependencies originated in a temporary rule. The register tracked the decision as a document. The environment experienced it as architecture.

NARRATIVE TURN / 02

THE ATTEMPTED CLEANUP

A new engineer eventually asked why the rule existed and proposed removing it. The change review produced immediate anxiety. One application owner remembered a nightly job. Another mentioned a vendor connection. A third pointed to a deployment template. Nobody could produce a complete list, so uncertainty became the strongest argument for preserving the status quo.

That moment exposed Inherited Wack in its purest form: nobody currently involved had made the original choice, everyone could name a reason not to touch it, and no one possessed enough evidence to design a safe exit. The exception had acquired Wack Gravity by becoming useful to things that arrived after its approval.

NARRATIVE TURN / 03

FROM RENEWAL TO RETIREMENT EXPERIMENT

The team changed the question. Instead of asking can we remove the rule, they asked what would we need to observe to remove it safely? They captured traffic for a bounded period, identified consumers, compared the live configuration with templates, interviewed owners, and tagged unknown flows. Each dependency received a disposition: required and redesign, obsolete and remove, unknown and investigate.

The exit plan became a sequence of reversible experiments. Narrow a source. Watch telemetry. Move one job. Validate. Narrow a destination. Validate again. The rule did not vanish in one heroic change window, but its radius shrank visibly. Every step replaced one reason for fear with one piece of evidence.

WHAT THE TEAM FINALLY SAW

Exception Creep is not simply an old approval. It is the transformation of a bounded deviation into an undocumented dependency platform. Dates alone cannot control it

because the real risk lives in propagation: new consumers, copied configuration, changed owners, and missing exit knowledge. An exception register becomes useful when it tracks current architecture, decision ownership, compensating activity, review triggers, and an executable path to a different state.

THE WAY OUT

WORK THE METHOD.

01 / FIND

Select the oldest or most frequently renewed exception and reconstruct its original promise. Compare the stated duration, systems, owners, and purpose with today's configuration and operating use.

02 / MAP

Observe actual flows, search templates and runbooks, identify every consumer, and mark unknown dependencies. Draw the exception as a path through architecture rather than a row in a governance table.

03 / EXPLAIN

For each dependency, name the current owner, purpose, necessity, and evidence. Distinguish fear of unknown impact from a demonstrated requirement to preserve the path.

04 / REDUCE

Build reversible experiments that shrink sources, destinations, permissions, consumers, or copied configurations. Give every experiment a success signal, rollback point, owner, and date.

05 / PROVE

Retain before-and-after maps, observation data, change records, validation results, residual dependencies, and the next review trigger. Evidence of safe reduction is more useful than another unchanged renewal.

FIELD NOTES

- An exception can expire on paper while continuing to grow in architecture.
- Uncertainty is a reason to instrument a change, not a permanent argument against one.
- Every renewal should compare current dependencies with the original approval.
- The safest exit is often a sequence of small experiments, not one dramatic removal.

WHAT CHANGED

After three cycles, the team removed the vendor source, moved two jobs, deleted the copied template rule, and constrained the remaining path to one documented service. The exception was still open, but it was no longer immortal. Its owner could show exactly what remained and the experiment that would address it next.

At the next review, the approval packet did not say removing this might break something. It showed what had depended on the rule, what no longer did, what the team had learned, and how the radius had changed. The conversation moved from inherited fear to managed retirement.

THE NEXT USEFUL QUESTION

TAKE THIS TO THE NEXT MEETING.

Take the oldest live exception and compare its current consumers, owners, and configuration with the original approval before renewing it again.

THREE LINES WORTH KEEPING

"Approved for fourteen days. Renewed for four years."

"The register tracked a document. The environment experienced architecture."

"Replace one reason for fear with one piece of evidence."

FIND / MAP / EXPLAIN / REDUCE / PROVE