

PAPER TIGER TECHNOLOGIES

Patch Management Policy

1. Purpose

The purpose of this Patch Management Policy is to keep operating systems, applications, and libraries in the PaperTiger environment current with security fixes. Timely patching closes known weaknesses before they can be exploited in the demonstration scenario.

Patching is the highest-volume and most reliable remediation action available, and it is also the most easily deferred. This policy exists to make the patch schedule explicit, to prioritize security fixes over feature updates, and to require verification so that a patch is known applied rather than assumed applied.

This policy is fictional and illustrative. It describes no real system, patch, or deployment.

Patching is the highest-leverage and least glamorous control in the program, so the policy makes it routine, measured, and impossible to silently skip.

The policy covers the awkward cases, zero-days and regressions and dependencies, because those are where a routine discipline silently fails.

Patching exists to close known weaknesses before they are exploited, and the policy exists to make that happen on a rhythm that balances speed against the risk of a bad fix.

Patching is a race against disclosure, and the policy makes the priority order, the test gate, and the verification step explicit so speed never becomes an excuse to skip the check.

2. Definitions

Term	Definition (illustrative)
Patch	A change that fixes a defect or security weakness in software.
Security patch	A patch that closes a known vulnerability.
Maintenance window	The scheduled period for applying patches.
Patchable inventory	The list of systems and libraries with their patch state.
Advisory	A notice of a newly disclosed vulnerability in a product or dependency.
Deferral	A documented decision to delay a patch with compensating controls.
Rollout	The staged application of a patch across

	systems.
Regression	A new defect introduced by a patch.
Verification	The confirmation that a patch is applied and the service healthy.
Emergency patch	A patch applied outside the window for a critical issue.
Patch coverage	The fraction of systems current on security patches.
Staged rollout	The phased deployment of a patch across a fleet.
Patching window	The scheduled interval during which routine patches are applied to a system.
Zero-day	A weakness that is exploited before a vendor patch exists.
Patch regression	A defect introduced by a patch that breaks existing behavior.
Patch window	The scheduled period in which routine patches are applied.
Patch testing	The confirmation that a fix works and does not break anything before production.
Change freeze	A period when routine changes pause to reduce risk.
Compensating control	A temporary measure that reduces risk while a patch is pending.
Patch deferral	The documented decision to delay a patch with its reason.
Patch verification	The confirmation that a fix applied and the weakness no longer reproduces.
Regression testing	The check that a patch did not break existing behavior.
Rollback plan	The prepared path to reverse a bad patch.
Security update	A fix released to address a weakness.
Hotfix	An urgent fix applied outside the routine window.
Patch compliance	The measure of systems that are current.
Patch integrity	The confirmation that a patch is untampered.
Rollout staging	The graduated deployment of a patch.
Patch exception	A documented decision to skip or delay a patch.
Compatibility matrix	A table showing which patch versions have been tested against each platform and the result.
End-of-support date	The date after which a vendor no longer provides security updates for a product version.

3. Scope and Boundaries

This policy applies to fictional systems, endpoints, containers, and third-party libraries inside the service boundary. It covers both operating-system and application patches.

It applies to every patch, whether routine or emergency, and to the decision to defer a patch as much as to apply it.

Out of scope are real systems and real patches. This policy creates no real obligation.

The scope includes third-party libraries on equal footing with operating systems, because a vulnerable dependency is a vulnerability in the product regardless of where the code came from.

Escalation for a patching issue follows a fixed chain: a system that falls behind goes to the patching administrator, then to the Information Security Lead when the weakness is exploited, and finally to the governance forum for a material exposure.

Patching covers every system in the demonstration, including internal tools, so an unpatched workstation cannot become the entry point that the servers were protected against.

Coverage is reported on a schedule; a system that falls behind is either patched or becomes a documented exception with a date.

Patching coverage is reviewed against the full inventory, not just the customer-facing systems.

Patching covers every system in the demonstration and is fictional.

A patch is not complete until it is verified, because a failed install reported as done is worse than an unpatched system.

This policy applies to security patches and functional updates for operating systems, applications, firmware, and third-party libraries across all environments.

4. Roles and Responsibilities

Role	Illustrative responsibility
Patch Management Lead	Owns the policy and the patch schedule.
System Administrator	Applies patches in the maintenance window.
Release Manager	Coordinates patch testing and rollout.
Risk Owner	Accepts risk when a patch is deferred.
Platform Team	Verifies service health after patching.
Trust Center Coordinator	Maintains the fictional label on shared copies.

The following RACI is illustrative.

Illustrative RACI for patching:

Activity	Patch Mgmt Lead	Sys Admin	Release Manager	Risk Owner
Identify patches	R	I	I	I
Test in staging	C	I	R	I
Apply in window	I	R	C	I
Verify	C	R	I	I
Defer	C	I	C	A

5. Policy Requirements and Controls

PTT-POL-012-C01. Maintain a patchable inventory

Systems and libraries are listed with their patch state, because you cannot patch what you do not know you have. The inventory is the input to every other step. The inventory is reconciled against scanning monthly.

PTT-POL-012-C02. Monitor patch sources

Vendor and dependency advisories are reviewed on a defined cadence so new security patches are identified promptly. A new notice is triaged within a day.

PTT-POL-012-C03. Prioritize security patches

Security fixes are ranked above feature updates, because closing a known weakness outranks delivering a new capability. A security patch is applied before a feature update.

PTT-POL-012-C04. Test before rollout

Patches are verified in staging before production, so a patch that introduces a regression is caught outside production. A patch that skipped testing is flagged.

PTT-POL-012-C05. Apply within the window

Patches are deployed within the illustrative timeline for their severity, and slippage is escalated. A missed window is logged with its reason.

PTT-POL-012-C06. Verify after deployment

The patch is confirmed applied and the service healthy, so a failed deployment is caught and remediated. The verification result is recorded.

PTT-POL-012-C07. Record deferrals

Skipped patches are logged with a reason and a new target date, so deferral is a decision, not an oversight.

PTT-POL-012-C08. Handle emergency patches

Critical patches may be applied outside the window through the emergency path and reviewed after the fact. A critical patch can deploy outside the window with the right approval.

PTT-POL-012-C09. Cover dependencies

Third-party libraries are patched or upgraded against advisories through the same process. Third-party components are patched, not just the operating system.

PTT-POL-012-C10. Rollback when needed

A patch that causes a regression is rolled back through the change process, and the rollback is recorded. A bad patch can be rolled back, so a fix does not become an outage.

PTT-POL-012-C11. Track patch coverage per system

Coverage is measured per system so that a fleet-level average does not hide a single unpatched high-value system. Per-system visibility drives targeted follow-up. Coverage is reported per system monthly.

PTT-POL-012-C12. Stage rollouts

Patches roll out in stages so a defect in the first wave is caught before it reaches the whole fleet. Staging limits the blast radius of a bad patch. The rollout is staged, so a problem is caught on a subset before it is broad.

PTT-POL-012-C13. Document the patch window

The maintenance window is documented so deployments happen at a known, low-risk time rather than whenever convenient. The window is documented, so teams know when patches run.

PTT-POL-012-C14. Verify patch integrity

Patches are verified to come from a trusted source before application, so a tampered patch is not introduced. The patch is checked for integrity, so a tampered patch is rejected.

PTT-POL-012-C15. Patch non-production first

Non-production environments are patched before production, so a problem is found in a lower-risk environment first. The patch runs in non-production first, so production is protected.

PTT-POL-012-C16. Patch on a defined cadence

Routine patches are applied within a scheduled window on a recurring basis, so systems do not fall months behind and the workload is predictable. The cadence is defined, so patching is a rhythm rather than an afterthought.

PTT-POL-012-C17. Handle zero-days with compensating controls

When no patch exists yet, the environment applies compensating controls such as isolation or monitoring, so the window between disclosure and patch is not an open door. A zero-day gets a compensating control while a fix is pending.

PTT-POL-012-C18. Detect patch regressions

A patch is watched for regression after rollout, and a regression triggers rollback, so a fix that breaks something is caught and reversed rather than left to fester. A regression introduced by a patch is caught and fixed.

PTT-POL-012-C19. Patch third-party components

The patching discipline extends to third-party components, not just the platform's own code, because a known weakness in a dependency is a known weakness in the environment. Third-party code is patched on the same discipline as the rest.

PTT-POL-012-C20. Patch the demonstration's own tools

The internal tooling is patched on the same discipline as the service, so a lagging internal tool does not become a back door. Internal tooling is included in the coverage report.

6. Operating Procedures

Identify available patches.

Advisories are reviewed and mapped to the patchable inventory.

Prioritize and schedule.

Security patches are scheduled by severity into the maintenance window.

Test in staging.

The patch is applied and verified in staging before production.

Deploy in the window.

The administrator applies the patch to production systems.

Verify.

Deployment is confirmed and service health is checked.

Log deferrals.

Any skipped patch is recorded with a reason and a new date.

Handle emergencies.

A critical patch is applied through the emergency path and reviewed after.

Report status.

Patch coverage and deferrals are reported on the defined cadence.

Report coverage.

Per-system patch coverage is reported on the defined cadence, and outliers are followed up.

Stage the rollout.

A patch is deployed to a first wave, verified, and then expanded to the remaining systems.

Verify a patch source.

The patch is verified to come from a trusted source before application.

Patch non-production first.

The patch is applied to non-production and verified before production.

Apply a routine patch.

The patch is tested on the subset, applied within the window, and the resulting state is recorded.

Apply an emergency patch.

The patch is applied on the accelerated path, the change is recorded, and it is folded back into the normal cadence.

Apply a compensating control.

The unpatched system is isolated or monitored, the control is recorded, and it is removed once the patch lands.

Deploy an emergency patch.

The change owner documents the urgency, the approver signs off, the patch is deployed, and the fix is verified after.

Report patch coverage.

The patch owner summarizes the coverage by system, and the Information Security Lead reviews the gaps with leadership.

Stage a rollout.

The patch owner deploys to a subset first, watches for regressions, and only then broadens the rollout.

Verify a patch.

The owner confirms the fix is applied and the vulnerability no longer reproduces.

Handle a zero-day.

The owner applies compensating controls, documents the gap, and fast-tracks the fix once one is available.

Run a regression test.

The owner tests the patch against existing behavior and records the result.

Execute a rollback.

The owner reverses a bad patch using the prepared plan and records the outcome.

Monitor patch sources.

The owner watches vendor and security notices and triages a new patch.

Verify patch integrity.

The owner checks the patch against its expected hash before applying it.

Document the patch window.

The owner records when patches run, so teams know the schedule.

Track patch exceptions.

The owner records a skipped or delayed patch with its reason and a new date.

Maintain a patch compatibility matrix.

The patch team records which patch versions are tested against each supported platform and marks combinations that are known to be incompatible before rollout.

Track vendor end-of-support dates.

The team maintains a list of every product version and its vendor end-of-support date so systems are upgraded before they stop receiving security patches.

Stage a patch rollout.

The patch team rolls a patch to a small group first, observes the result, then expands to the remaining systems in stages to catch problems before full deployment.

7. Frequency and SLAs

Illustrative patch cadence:

Activity	Frequency (illustrative)
Advisory review	Weekly
Maintenance window	Monthly
Critical security patch	Emergency or same-cycle
Deferral review	Monthly

8. Exceptions and Lifecycle

An exception permits deferring a patch when applying it would break a dependency, and the Patch Management Lead documents the compensating control and a retry date.

Deferrals are reviewed monthly in the demo, and a patch deferred across two cycles is escalated rather than deferred again.

A patch deferred across two cycles is escalated rather than deferred again, because a repeated deferral usually means the compensating control is not actually compensating.

9. Records and Evidence

Record type	What it demonstrates (illustrative)
Patch inventory	The systems and libraries with patch state.
Advisory review log	The reviewed patch sources.
Test results	The staging verification.
Deployment records	The applied patches.
Verification records	The post-deployment health check.
Deferral log	The skipped patches with rationale.
Coverage report	The per-system patch state.
Rollout staging	The phased deployment plan and results.
Patching schedule	The cadence and windows for each system.
Compliance records	The applied and outstanding patches per system.
Patch deployment records	The dated records of patches applied and verified.
Regression test results	The recorded outcomes of patch testing.
Freeze calendar	The published freeze windows.
Compliance report	The measured patch coverage per system.
Hotfix records	The documented urgent fixes.
Compatibility matrix	The tested patch-to-platform combinations and any known conflicts.

End-of-support register	Lists each product version with its end-of-support date and upgrade plan.
Rollout staging record	Shows the stages of a patch release and the result at each stage.

10. Oversight and Metrics

- Patch coverage: the fraction of systems current on security patches (illustrative).
- Time to patch: within the illustrative severity window.
- Deferral count: trending down.
- Regression rate: patches rolled back trending toward zero.
- Per-system coverage: no high-value system outside the security patch window.
- Patch coverage: the share of systems current within the window trending to 100 percent.
- Patch lag: the median days a system runs unpatched trending down.
- Percentage of systems within the patch window.
- Time from patch release to verified deployment for critical fixes.
- Time from release to verified deployment for critical fixes.
- Regressions introduced per hundred patches.
- Emergency patches per period.
- Systems out of compliance.
- Hotfixes per period.

11. Appendix

Illustrative patch timelines:

Patch type	Timeline (illustrative)
Critical security	Emergency or same-cycle
High security	Next maintenance window
Routine	Scheduled cycle
Deferred	Documented with compensating control

12. Related Policy Dependencies

- Vulnerability Management Policy (identifies the findings patches close)
- Secure Configuration Policy (the baseline patches preserve)
- Asset Management Policy (the inventory patches depend on)
- Change Management Policy (governs patch deployments)
- Vulnerability Management Policy (identifies the patches this policy applies)

13. Review Cadence

The Patch Management Lead reviews this policy at least annually, after a change to the illustrated environment, and after a significant simulated patching failure.

Revision History

Version	Date	Change summary	Author
1.0	September 28, 2026	Original enterprise-depth fictional demo policy for controlled trust-center testing.	Paper Tiger Technologies

Document Control

Field	Value
Document ID	PTT-POL-012
Document Owner	Patch Management Lead
Status	Fictional demo artifact
Effective Date (illustrative)	September 28, 2026
Review Cadence	At least annually and after material fictional changes
Classification	Internal Demo
Approval	Illustrative governance approval only; no real adoption represented