Impact of Red Hat fapolicyd on InterSystems IRIS



RELEASE:
SEPTEMBER 2026

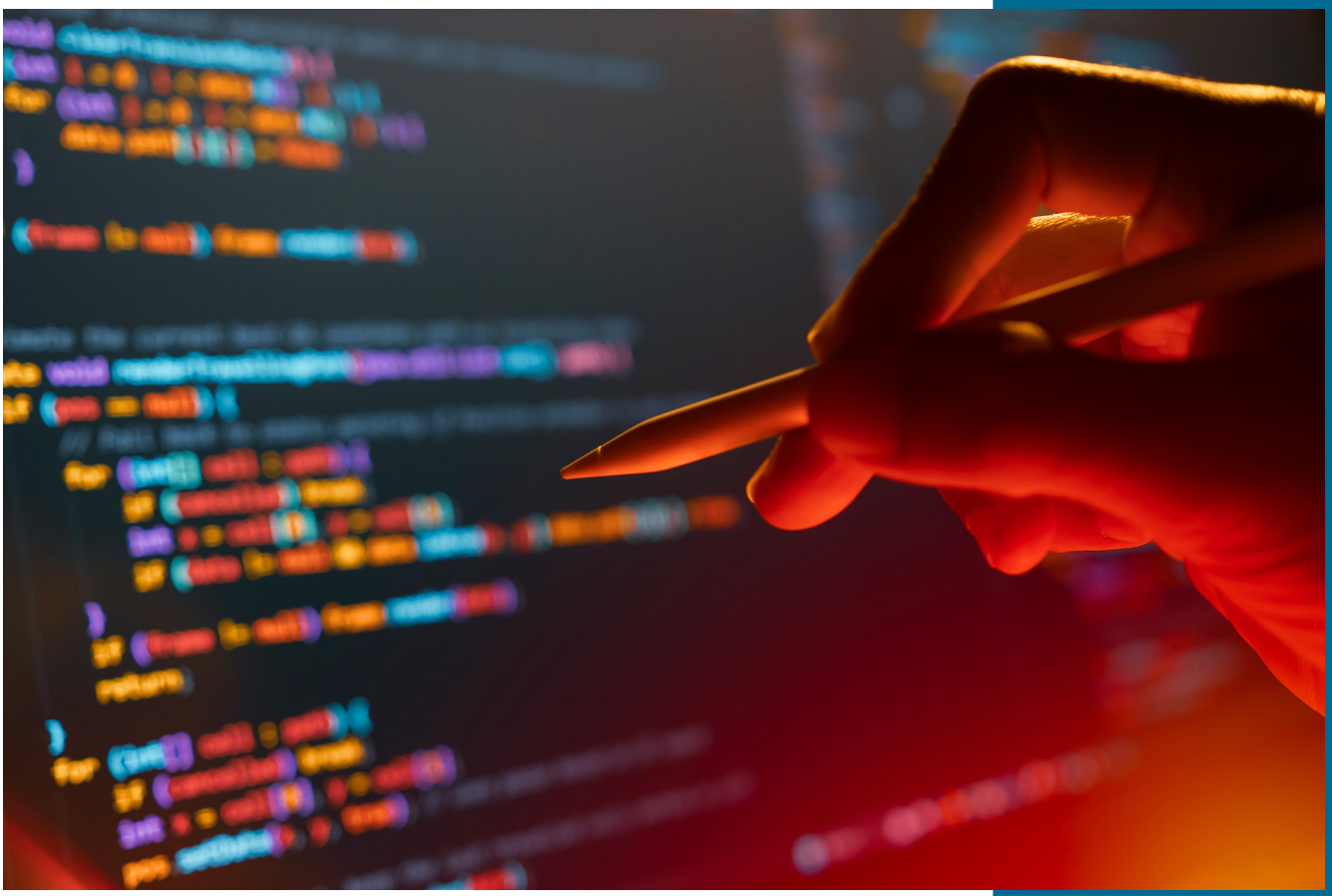
InterSystems®
Creative data technology

Table of Contents

A Security Benchmark Report	3
Executive Summary	4
InterSystems IRIS Under Default and Reconfigured `fapolicyd`	4
`fapolicyd` and the Trust Model	5
InterSystems IRIS Behavior Under Default `fapolicyd` Policy	5
Approach to a Validated Trust Solution	5
Test Environment and Methodology	6
Test Results and Findings	7
Security Considerations and Trade-offs	8
SELinux Interaction	8
Containerization Considerations	8
Deployment Guidance	9
Conclusion	9
Appendix	10-11

A Security Benchmark Report

This report examines the incompatibility between **InterSystems IRIS** and the default enforcement of Red Hat's File Access Policy Daemon (`fapolicyd`), which restricts execution to trusted RPM-installed software. Under default `fapolicyd` protocols, installation of InterSystems IRIS loads untrusted shared libraries and fails to initialize. This report details a validated solution using custom `fapolicyd` rules and trust configurations, enabling normal InterSystems IRIS operation without denials. These findings underscore the importance of a staged deployment approach, including an initial validation in Permissive mode to check for denials, before moving on to Enforcing, with trust, to ensure both operational continuity and security.



Executive Summary

Modern Linux systems increasingly rely on application allowlisting to defend against supply chain attacks, unauthorized binaries, and runtime exploitation techniques. On Red Hat Enterprise Linux (RHEL) systems, the File Access Policy Daemon (`fapolicyd`) is a key control that enforces a trust-based execution model, restricting applications and libraries not explicitly permitted by policy. This approach helps mitigate common attack techniques described in MITRE ATT&CK, such as unauthorized execution, execution flow hijacking, and untrusted code transfer.

This trust-based model introduces incompatibilities with software not distributed through standard package management systems such as RPM. InterSystems IRIS falls into this category. When `fapolicyd` is enabled in its default Enforcing configuration, InterSystems IRIS cannot be initialized because its binaries and shared libraries are not trusted by the operating system. In practice, **InterSystems IRIS does not function under the default `fapolicyd` policy (which trusts only RPM-installed software), but operates normally once appropriate trust rules are configured.**

Extensive testing across multiple environments and more than 40 hours of validation demonstrates that this incompatibility is not a platform limitation, but a configuration issue. With appropriate `fapolicyd` rules and trust database configuration, InterSystems IRIS operates normally with no denials observed across 13 operational categories and more than 25 test phases. This paper presents the necessary rules and configurations.

InterSystems IRIS Under Default and Reconfigured fapolicyd

InterSystems IRIS is incompatible with unconfigured `fapolicyd` enforcement (which relies on RPM-based trust by default), it is fully compatible with properly configured policies.

Application control has become a foundational element of modern Linux security. Rather than relying solely on traditional access controls, systems increasingly enforce trust-based execution policies that determine which binaries and libraries are allowed to run. This approach mitigates risks such as unauthorized software execution, persistence mechanisms, and exploitation of compromised environments.

Red Hat Enterprise Linux implements this model through `fapolicyd`, which evaluates file access events against ordered rules and trust relationships. Because InterSystems IRIS is distributed as a `tar`-based installer rather than an RPM package, its components are not trusted by default, resulting in immediate operational failure when enforcement is enabled.

This security report provides background on the `fapolicyd` trust model, analyzes the problem observed with InterSystems IRIS under default policy, presents a validated solution with supporting test results, and discusses deployment and security considerations.

fapolicyd and the Trust Model

Unlike SELinux or AppArmor, which enforce label-based mandatory access control, `fapolicyd` focuses on execution allowlisting by controlling which binaries and libraries are permitted to run based on trust.

With `fapolicyd`, files installed via RPM are automatically trusted, while all other files are treated as untrusted unless explicitly allowed. `fapolicyd` uses the Linux fanotify API to monitor file access events and applies ordered rules to determine whether operations are permitted. A critical aspect of the default policy is that untrusted shared libraries are denied.

Because modern applications depend heavily on dynamic linking, this behavior has significant implications for software installed outside standard package managers.

InterSystems IRIS Behavior Under Default fapolicyd Policy

The trust-based execution model and default shared library restrictions directly impact how InterSystems IRIS behaves under the default `fapolicyd` policy, where InterSystems IRIS does not function.

This failure occurs because the default policy denies untrusted shared library loading, which InterSystems IRIS requires for runtime operation. When `fapolicyd` is enabled in enforcing mode, InterSystems IRIS fails immediately due to three primary factors:

- InterSystems IRIS is not installed via RPM and is therefore not trusted
- The installation process copies files into new locations, invalidating any existing trust
- The default policy explicitly denies access to untrusted shared libraries

As a result, InterSystems IRIS cannot initialize, and no database functionality or InterSystems IRIS-based applications are available.

Approach to a Validated Trust Solution

The incompatibility between InterSystems IRIS and `fapolicyd` can be resolved through explicit trust configuration and policy adjustments.

`fapolicyd` evaluates rules in priority order. Custom rules must be assigned lower numeric prefixes (for example, 22-) than default policies (typically 41-) to override deny rules. Custom rules allow access to InterSystems IRIS installation directories and enable runtime linker behavior required for shared library loading. In parallel, key InterSystems IRIS binaries and directories are added to the `fapolicyd` trust database (see Appendix C).

The validated solution combines rule-based allowances with configuration of the `fapolicyd` trust database. It can be implemented using either a broad (permissive) rule set or a more restrictive minimal rule set (see Appendix A and Appendix B).

Both approaches were validated and resulted in identical functional outcomes, with no `fapolicyd` denials observed. The minimal rule set reduces the allowed execution surface by more explicitly scoping permissions.

With these configurations in place, `fapolicyd` transitions from blocking InterSystems IRIS to enforcing a controlled and auditable execution model in which InterSystems IRIS operates without restriction.

Test Environment and Methodology

The validation approach followed a structured progression through four phases: Baseline, Permissive, Enforcing (Default), and Enforcing (Trusted). Testing was conducted across RHEL 8 and RHEL 10 on both x86_64 and ARM64 platforms, covering InterSystems IRIS versions from 2023.1 through 2026.1. Across these environments, 13 operational categories were evaluated, including installation, upgrades, database operations, web services, file I/O, and external execution. Each phase validated both functionality and absence of policy denials. The rule sets used in testing are provided in Appendix A and Appendix B.

The following table summarizes the observed behavior across all test phases and provides a consolidated view of both coverage and outcomes, which are discussed in detail in the following section.

Category	Baseline	Permissive	Enforcing (Default)	Enforcing (Trusted)
Installation & Upgrade	Pass	Pass	Fail	Pass
Startup/Shutdown	Pass	Pass	Fail	Pass
Database I/O & Compilation	Pass	Pass	N/A (blocked)	Pass
Web Services (CSP/SuperServer)	Pass	Pass	N/A (blocked)	Pass
File Operations (Backup/Journal/ Temp)	Pass	Pass	N/A (blocked)	Pass
External Execution ($ZF/Scripts)	Pass	Pass	N/A (blocked)	Pass
`fapolicyd` Denials	N/A	Logged Only	Blocking	None

Table 1: InterSystems IRIS Phased Test Coverage Overview and Results

Test Results and Findings

The following figure shows a simplified view of the execution flow for InterSystems IRIS processes under `fapolicyd` enforcement. It illustrates how file access requests are intercepted, evaluated for trust, and processed through rule evaluation before resulting in either an allow or deny decision. This flow highlights the exact point at which default policy behavior blocks InterSystems IRIS and where custom rules alter the outcome.

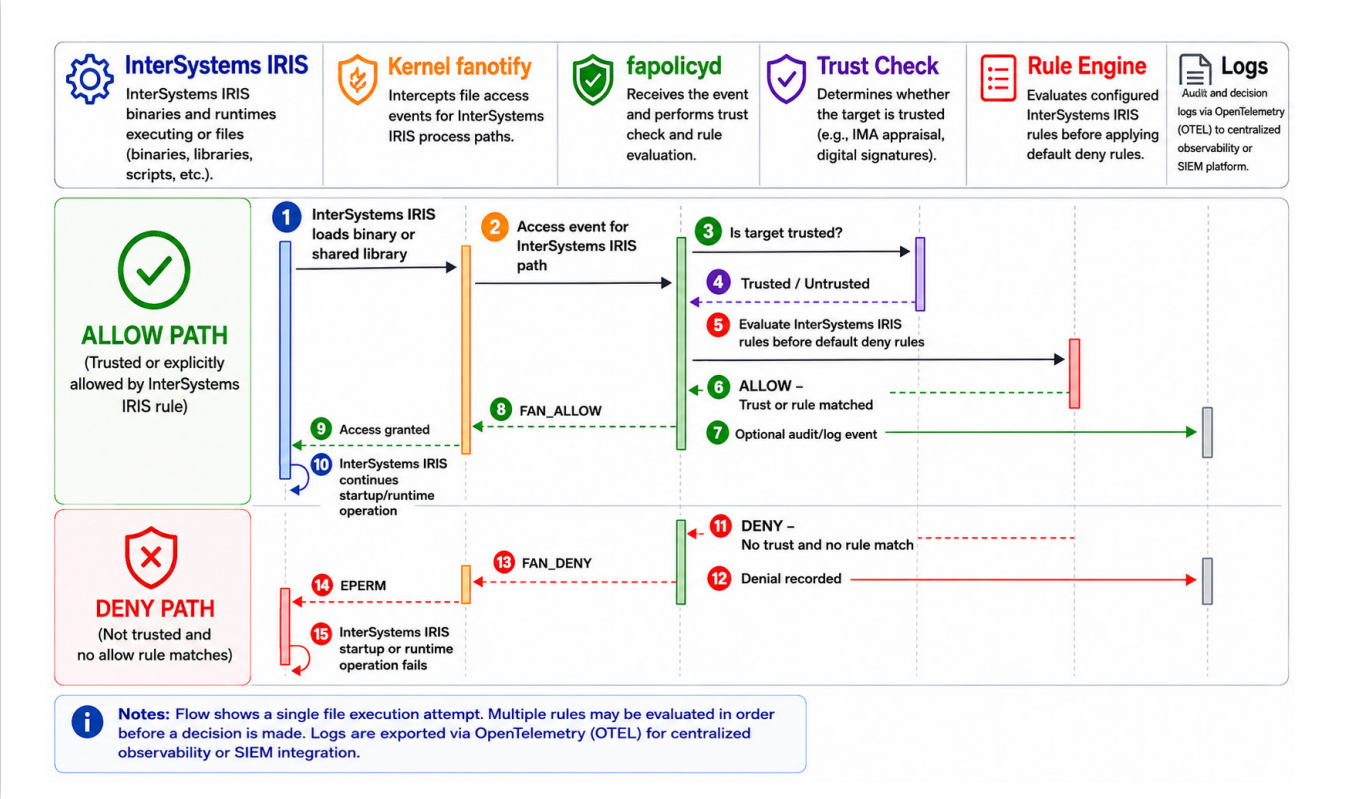


Figure 1: Simplified `fapolicyd` *Execution Flow for InterSystems IRIS Processes*

As shown in the figure, the denial occurs during shared library evaluation when the target is not trusted and no allow rule applies.

Results were consistent across all tested environments. In Baseline and Permissive modes, InterSystems IRIS operated normally. In Enforcing mode with default policies, InterSystems IRIS failed completely. When custom trust rules were applied (Enforcing mode, Trusted), InterSystems IRIS operated successfully with no denials observed. No measurable performance impact was identified, and upgrade scenarios preserved data integrity and operational continuity.

Security Considerations and Trade-offs

Residual Risk Model: This configuration assumes standard Linux file permissions are correctly enforced and that the host is not compromised at the root level. These controls are designed to manage execution of untrusted code, not to contain a fully privileged attacker.

The rule sets required for InterSystems IRIS compatibility grant access to InterSystems IRIS directories and enable dynamic library loading. The minimal rule set reduces this access by scoping permissions more explicitly, but it does not eliminate execution risk within trusted paths.

Adversarial testing confirms expected behavior, such as allowing execution within trusted directories. This reinforces the importance of a defense-in-depth strategy. Organizations should combine `fapolicyd` with SELinux, standard file permissions, and monitoring controls. Additional protections such as IMA/EVM and EDR/SIEM platforms can further strengthen security posture.

Forwarding `fapolicyd`, SELinux, and InterSystems IRIS operational logs to a centralized observability or SIEM platform using OpenTelemetry (OTEL) is recommended for audit visibility.

SELinux Interaction

In environments where SELinux is enabled, both SELinux and `fapolicyd` must allow an operation for it to succeed.

Operationally, administrators should monitor both SELinux audit logs and `fapolicyd` logs during deployment. As a practical troubleshooting step, if an operation fails, check both AVC (SELinux) and FANOTIFY (`fapolicyd`) logs to identify the source of denial.

Containerization Considerations

`fapolicyd` cannot operate within container environments because it relies on kernel-level fanotify mechanisms and system-level daemon capabilities. Host-level `fapolicyd` does not enforce policy inside containers.

As a result, this approach applies to host-based deployments such as bare metal and virtual machines. Containerized environments must rely on alternative security mechanisms such as SELinux, seccomp, or platform-native controls.

Deployment Guidance

Organizations should follow a staged deployment approach. Initial validation should be performed in permissive mode to observe potential denials without impacting functionality. Once rules are confirmed, Enforcing mode (Trusted) can be enabled safely.

Important: Never switch to Enforcing mode before validation, as this may result in an outage of InterSystems IRIS-based applications.

If `fapolicyd` is not enabled, no action is required. If it is enabled in permissive mode, InterSystems IRIS will function normally but policies are not enforced. If enforcement is required, custom trust rules must be deployed prior to installation or startup (see Appendix A and Appendix C). Additional implementation considerations and common pitfalls are summarized in Appendix D.

Conclusion

`fapolicyd` provides a strong mechanism for controlling application execution and aligns with MITRE ATT&CK mitigations and NIST SP 800-53 CM-7 by enforcing allowlisting.

Although default policies are incompatible with InterSystems IRIS, this limitation is fully resolved through proper configuration. With validated rules in place, InterSystems IRIS operates reliably with no functional or performance impact, supporting secure deployment within a defense-in-depth model.

In summary,

- Default `fapolicyd` enforcement blocks InterSystems IRIS completely.
- Proper configuration enables full compatibility.
- Both broad and minimal rule sets are validated and supported.
- Minimal rules improve security posture but do not eliminate all risk.
- Container environments require alternative approaches.
- Always validate in Permissive mode before enabling enforcement.

Appendices

Appendix A: Validated `fapolicyd` Rules (Broad)

Warning: Validate in Permissive mode before enforcement.

```
allow perm=any uid=0 : dir=/usr/InterSystems IRISsys/

allow perm=any uid=InterSystems IRISusr : dir=/usr/
InterSystems IRISsys/ allow perm=any uid=InterSystems
IRISowner : dir=/usr/InterSystems IRISsys/

allow perm=any uid=0 pattern=ld_so : all

allow perm=any uid=InterSystems IRISusr pattern=ld_so : all
allow perm=any uid=InterSystems IRISowner pattern=ld_so : all
```

Appendix B: Minimal (Least-Privilege) Rules

Note: These rules reduce scope but are not exhaustive. Additional restrictions may be possible depending on deployment requirements.

```
allow perm=execute uid=0 : dir=/usr/InterSystems IRISsys/bin/

allow perm=execute uid=InterSystems IRISusr : dir=/usr/
InterSystems IRISsys/bin/ allow perm=execute uid=InterSystems
IRISowner : dir=/usr/InterSystems IRISsys/bin/

allow perm=any uid=0 pattern=ld_so : dir=/usr/InterSystems
IRISsys/lib/

allow perm=any uid=InterSystems IRISusr pattern=ld_so : dir=/
usr/InterSystems IRISsys/lib/ allow perm=any uid=InterSystems
IRISowner pattern=ld_so : dir=/usr/InterSystems IRISsys/lib/
```

Appendix C: Trust Commands

```
fapolicyd-cli --file add /usr/bin/InterSystems IRIS --trust-file
InterSystems IRISsys fapolicyd-cli --update


fapolicyd-cli --file add /usr/InterSystems IRISsys/ --trust-file
InterSystems IRISsysman fapolicyd-cli --update
```

Appendix D: Implementation Notes and Considerations

The following notes summarize key operational considerations identified during testing and deployment.

Rule Priority

- `fapolicyd` evaluates rules in numerical order.
- Custom InterSystems IRIS rules must use a lower numeric prefix (for example, 22-) than default policies (typically 41-) to ensure they are evaluated first.
- Incorrect rule ordering is a common cause of unexpected denials.

Path Sensitivity

- `fapolicyd` trust and rules are path-based.
- InterSystems IRIS installation paths may vary by version and deployment method.
- Always verify and update rule paths to match the actual installation (for example, `/usr/InterSystems IRISsys/`).

Trust Database Scope

- Trust entries must be explicitly added for InterSystems IRIS binaries and directories.
- Trust does not automatically propagate when files are copied during installation.
- Re-run `fapolicyd-cli --update` after modifying trust entries.

Permissive Mode Validation

- Always validate configurations in permissive mode before enabling enforcement.
- Permissive mode allows operations while logging what would have been denied.
- This step is critical to avoid production outages.

Monitoring and Troubleshooting

- Review `fapolicyd` and SELinux logs during initial deployment.
- Denials may originate from either control mechanism.
- Integrating logs into centralized observability platforms (for example, via OpenTelemetry) improves visibility.

---

[1] A. Sedgwick et al., Guide to Application Whitelisting (NIST Special Pub. 800-167, 2015), https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-167.pdf

[2] https://attack.mitre.org/

[3] Joint Task Force, Security and Privacy Controls for Information Systems and Organizations (NIST SP 800-53 Rev. 5, 2020), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf



Products and services are subject to InterSystems terms and conditions. Information is provided for general informational purposes, may change without notice, and is provided without warranty. Product availability, functionality, and regulatory status may vary by offering, jurisdiction, and intended use.

For information regarding specific offerings, including availability and regulatory status, please contact your InterSystems Representative.

For information regarding InterSystems approach to AI, visit ***InterSystems AI Guidelines*****.**

© 2026 InterSystems Corporation. All rights reserved. For trademark information, see **InterSystems Notices, Terms & Conditions**. 0917 26