Automated Isolation of Vulnerable Workloads

Aiden Johnson
HPC-SYS, Los Alamos National Laboratory
B.S. Computer Science, The Pennsylvania State University
Los Alamos, New Mexico, USA
Mentor: Thomas Mackell, HPC-SYS

***Abstract*—Kubernetes and OpenShift support scientific services, data platforms, and artificial-intelligence workflows at high-performance computing facilities. Red Hat Advanced Cluster Security (RHACS/StackRox) identifies vulnerable images, policy violations, and risky deployments, but operators still must interpret risk and coordinate containment. That handoff is slow; immediate deny-all isolation can interrupt shared services before owners remediate. We present an automated, risk-based system that converts RHACS findings into controlled Kubernetes NetworkPolicy actions. A shared backend classifies risk, assigns configurable remediation windows, stores workload state, and applies or removes managed policies. Dashboard, CLI, operator, REST API, and Splunk integrations support manual and automated response. Namespace scoping, policy simulation, rollback, opt-out metadata, and structured audit events preserve operational control. Prototype validation demonstrates end-to-end detection, tracking, containment, and recovery for representative vulnerable workloads. Together, these mechanisms make response timing visible, repeatable, and recoverable instead of relying on ad hoc operator judgment.**

***Index Terms*—Kubernetes, RHACS, NetworkPolicy, vulnerability response, workload isolation, HPC security**

I. Motivation and Contributions

Container platforms enable portable scientific applications and operational services, yet shared ownership complicates vulnerability response. Security teams may know that a workload is high risk while application owners need time to test an image update or configuration change. Without a controlled containment path, safely supporting user-provided container environments on shared infrastructure remains difficult. The question is whether containment can be automated without becoming opaque or irreversible.

The system contributes a reusable isolation engine that consumes RHACS risk data and Kubernetes metadata; risk-level remediation windows that separate detection from enforcement; managed ingress-and-egress NetworkPolicies with explicit removal paths; workload metadata recording risk, state, timing, and operator opt-out; and multiple interfaces over one backend so manual review, continuous reconciliation, and external automation produce consistent actions. Persisting state on workloads also preserves decisions across controller restarts and interface changes.

II. Architecture and Workflow

RHACS supplies deployment identity, namespace, cluster, image vulnerabilities, policy violations, and aggregate risk scores. The detector filters excluded namespaces, resolves the Kubernetes workload, and classifies each finding using configurable thresholds. The isolation engine coordinates policy generation, state transitions, application, and cleanup. Figure 1 summarizes the shared architecture.

The workflow has five stages:

1) **Detect:** Retrieve current findings from RHACS.
2) **Classify:** Map each workload to a risk level and remediation window.
3) **Track:** Write metadata to the owning Deployment, DaemonSet, StatefulSet, ReplicaSet, Job, or CronJob.
4) **Wait:** Allow time for remediation before enforcement.
5) **Isolate or recover:** Apply a managed deny-all NetworkPolicy after the remediation window expires, or remove managed isolation and stale tracking data when the risk is resolved.

Continuous reconciliation repeats this decision during every scan. The default remediation windows are 15 days for critical scores above 40, 21 days for high scores from 30 through 40, and 45 days for medium scores from 10 to below 30. Thresholds and windows are configurable. Because the default tracking threshold is 30, medium-risk tracking is enabled only when administrators lower that threshold below 30.

III. Safety, Policy Semantics, and Recovery

Kubernetes NetworkPolicies are additive: an allow rule in another matching policy may continue to permit traffic. The implementation treats compatibility as a first-class concern. It labels isolated live pods and recommends excluding that label from broad namespace allow policies. Live pod labels are patched without changing templates, avoiding an application rollout during containment. Operators must also account for DNS, health checks, service-mesh sidecars, container network interface behavior, and other required paths before strict isolation.

Safeguards include always-excluded system namespaces, configurable allowed and ignored namespaces, policy preview and RHACS simulation for interactive use, dry-run REST actions, direct removal of managed policies through the Kubernetes API, and an automatic-isolation opt-out annotation for deliberately restored workloads. Structured events and metadata retain the reason, timing, and outcome of each decision for audit and troubleshooting.

LA-UR-26-26453

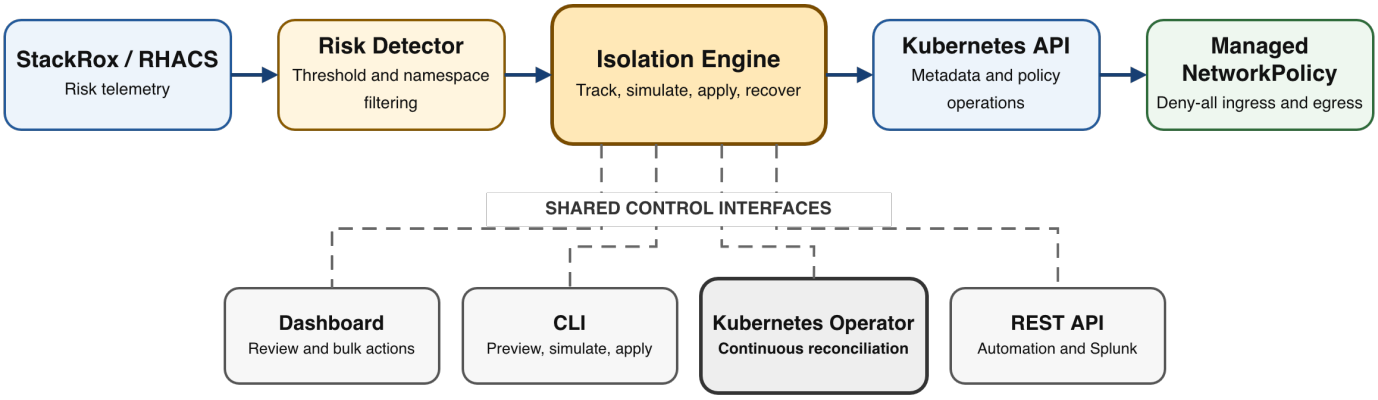


Fig. 1. System architecture of the automated isolation platform. RHACS findings pass through a shared risk detector and isolation engine; four interfaces coordinate enforcement through the Kubernetes API and managed NetworkPolicies.

IV. Prototype Results and HPC Relevance

The prototype implements the complete control path from RHACS findings to Kubernetes containment. Functional tests with representative vulnerable workloads verified risk filtering, scheduled isolation, managed policy creation, workload and pod labeling, explicit removal, opt-out behavior, and cleanup after risk resolution. State is exposed through a Streamlit dashboard, CLI, operator, REST API, and Splunk dashboards and alert actions. A shared engine reduces divergence between an analyst's manual action and the operator's automated action. By making containment controlled and reversible, the system moves HPC centers closer to safely supporting user-provided container environments.

For HPC centers, the approach provides repeatable response for Kubernetes-hosted scientific portals, data services, workflows, and AI support infrastructure. It does not replace patching or ownership processes; it makes the interval between discovery and remediation explicit, observable, and enforceable. Current evidence is functional rather than a large-scale performance study. Planned evaluation will measure scan time and isolation latency as workload counts increase, policy-application and rollback success, and operational impact on common cluster dependencies.

V. Conclusion

By joining RHACS risk signals with configurable time windows, Kubernetes-native enforcement, shared interfaces, and auditable recovery, the system closes the gap between vulnerability detection and containment while preserving safeguards required in shared computing environments.

Acknowledgment and AI Use Disclosure

The author thanks mentor Thomas Mackell and the HPC-SYS team. This work was performed at Los Alamos National Laboratory under the 2026 Cybersecurity Student Program. OpenAI ChatGPT, GPT-5.6 Pro (accessed July 17, 2026), was used for writing and editing assistance. The author reviewed the resulting text and retains responsibility for the submission.