International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V73I5P115

Volume 73 Issue 5, 113-124, May 2025

Original Article

Comparative Performance Evaluation of Modern Container Security Agents: Red Hat ACS, Wiz, SentinelOne, and Tenable

Harikishore Allu Balan[1], Bikash Agarwal[2]

[1,2]*T-Mobile, Principal Solution Architect, WA, USA.*

[1]*Corresponding Author : harikishore.allubalan@ieee.org*

Received: 29 March 2025 Revised: 02 May 2025 Accepted: 16 May 2025 Published: 30 May 2025

Abstract *- Containerized microservice applications have become the central design entity for how modern development and operations teams build and deploy software. Robust, configurable, and adaptable security agents are important in securing the applications. This article offers a closely monitored study with a detailed examination of four widely implemented container security platforms—Red Hat Advanced Cluster Security (ACS), Wiz, SentinelOne, and Tenable. Unlike off-shelf comparisons by third-party agents, our analysis is grounded in the practical development and deployment of the agents with realistic user traffic environments. The evaluation of each security agent's capabilities in handling vulnerabilities like threat detection, runtime defence, policy enforcement, and deployment pipeline integration are continually measured and compared. While the security agents met all basic security expectations for our study, we had to consider voice applications and how they balance operational efficiency, deployment complexity, and overall protection strategy. This comparative insight will help organizations like ours select a solution aligned with their specific cloud-native architecture and security posture.*

Keywords *- Container Security, Kubernetes, Red Hat ACS, Wiz, SentinelOne, Tenable, DevSecOps.*

1. Introduction

The evolution of applications towards containerization and Kubernetes as a runtime orchestrator for PODs has reshaped how modern organizations build, deploy, and scale their applications. The cloud platform offers faster iteration cycles, workload portability, and operational consistency across environments. However, they also introduce new and complex security challenges, often misaligned with organizational standards. The dynamic, fully managed containers, with their short lifespans and the distributed nature of Kubernetes clusters, create gaps that traditional security tools are not equipped to handle effectively.

Despite the industry growth towards container adoption growing faster and faster, there remains a clear research and implementation gap around how runtime threats are detected and managed in real-world environments. Many organizations still partially rely on static scanning or configuration checks and are more often reactive to security incidents, which, while important, are insufficient to meet current cloud requirements to monitor the threats and lateral movement within a cluster actively. Moreover, the rapid pace at which DevOps models are being implemented has outstripped the capabilities of legacy security models to provide meaningful, actionable insights during execution. This paper focuses on finding and addressing these gaps by evaluating and comparing four widely used container security platforms—Red Hat Advanced Cluster Security (ACS), Wiz, SentinelOne, and Tenable. The real-world simulation models document the investigation and analysis of how each platform performs across core areas such as live threat detection, runtime enforcement, compliance integration, and operational efficiency. Our goal is to help security practitioners understand the importance of application design concerning security agents' coexistence, where each tool fits in modern DevSecOps workflows, and how well it balances visibility, protection, and performance in cloud-native infrastructure.

2. Methodology

To thoroughly evaluate the effectiveness of today's leading container security solutions, we went beyond vendor claims and marketing whitepapers. While most tools attempt to position themselves as the most capable, their performance is often judged under artificial or narrowly defined conditions. In contrast, our approach involved

deploying these solutions in a hybrid testbed that mirrors the complexity of real-world cloud-native systems.

Our environment combined AWS-hosted Kubernetes clusters and on-premises infrastructure to replicate production-grade workloads. This included microservices communicating over service meshes, APIs exposed to external clients, and workloads subject to continuous integration and delivery pipelines. Each security agent—Red Hat ACS, Wiz, SentinelOne, and Tenable—was evaluated under steady-state and adversarial scenarios.

A diverse set of simulated threat scenarios was carefully designed and executed to assess how each security platform performs under realistic attack conditions. These included container breakout attempts, privilege escalation exploits, reverse shell injections, and deployment of known malicious payloads. The team leveraged well-established industry tools such **as** kube-hunter and kube-bench and custom scripts aligned with the MITRE ATT&CK framework for containerized environments.

What sets this evaluation apart is that the testing was conducted with near-production user traffic. Rather than isolating the test conditions in a lab, the threats were introduced during active user session emulation—mirroring real-world operational complexity. The test ensured that the results focused on how well each agent detected and responded to attacks and their effects on the underlying application system under realistic workload pressure.

2.1. Five Critical Performance Dimensions

2.1.1. Detect and Visibility

Agents' runtime behavior, anomalies, and misconfiguration possibilities.

2.1.2. Runtime Protection

Real-time threats such as SSH privilege access escalation and container drift with privilege escalation.

2.1.3. Compliance Check

Agents support key security standards, primarily CIS Benchmarks, NIST, and PCI-DSS.

2.1.4. DevSecOps with CI/CD Integration

Agent support to CI/CD workflows, infrastructure as code (IaC), and GitOps practices. Performance Overhead – What is the impact on CPU and memory resources at both the pod and node levels?

2.1.5. Performance Overhead

What is the impact on CPU and memory resources at both the pod and node levels?

The test ensured that the results focused on how well each agent detected and responded to attacks and their effects on the underlying application system under realistic workload pressure.

Five critical performance dimensions:

Detect and Visibility – Agents' runtime behavior, anomalies, and misconfiguration possibilities.

Runtime Protection – Real-time threats such as SSH privilege access escalation and container drift with privilege escalation.

Compliance Check – Agents support key security standards, primarily CIS Benchmarks, NIST, and PCI-DSS.

DevSecOps with CI/CD Integration – Agent support to CI/CD workflows, infrastructure as code (IaC), and GitOps practices. Performance Overhead – What is the impact on CPU and memory resources at both the pod and node levels?

The approach to the research and analysis work established in this paper is its grounding in a practical deployment scenario. For example, Use of reverse shell attempts to each worker instance, where SentinelOne very successfully intercepted reverse shell attempts within a second, while Red Hat ACS was seen to have more visibility into privilege escalation at the admission control layer through enforced policies. Agentless solutions, like Wiz, showed strong trends in the visibility of threats but fell short in runtime enforcement when rapid response was needed. Based on the other detailed findings, the paper underscores the importance of selecting the right security tool for an application-specific design approach based on features and how it handles realistic load with real risk. For cloud-native environments—especially those managing sensitive data or exposed to public access—agents with runtime visibility and automated response mechanisms offer a clear operational advantage.

3. Industry-standard Security Agents for Container Platforms

3.1. Red Hat Advanced Cluster Security (ACS)

Red Hat ACS is a native solution from Kubernetes that offers strong runtime threat detection and policy enforcement features. The agent platform integrates seamlessly with the Red Hat OpenShift cloud provider platform and is compatible with upstream Kubernetes environments. The ACS depends primarily on technologies like eBPF and Kubernetes admission controllers, enabling it to track container activity and intervene in potentially harmful operations before they materialize into threats. The solution is more tailored to suit multiple applications and

supports customizable policy frameworks, allowing teams to govern workload behavior precisely. The minimum configuration for ACS is 200 millicuries of CPU and 256 MiB of memory per instance, making it a practical fit for production clusters concerned with stability and overhead.

3.2. Wiz

Wiz, on the other hand, takes a markedly different approach by offering agentless cloud security. Rather than running inside the cluster, Wiz connects via cloud provider APIs to scan for misconfigurations, compliance risks, and potential vulnerabilities across infrastructure layers within the cluster. This agentless design makes Wiz particularly well-suited for organizations operating in multi-cloud environments or seeking to reduce deployment friction. Although it does not provide real-time runtime enforcement, its strength lies in broad, cross-environment visibility. Wiz delivers comprehensive context with a minimal footprint that imposes the need for very low demands on CPU and memory, making it ideal for security-aware but performance-sensitive environments.

3.3. SentinelOne

SentinelOne is an agent-based endpoint security solution requiring a dedicated Kubernetes domain resource through an agent-based deployment model. The core functionality mainly centres around real-time behavioral analysis using machine learning and eBPF-based kernel system space tracing. The agent can be configured to have proactive monitors running on the container continuously and reacting to threats autonomously, offering a robust layer of defence against emerging and evasive threats. This capability demands a higher cost of operation, whereas the SentinelOne agent requires up to 1 vCPU and approximately 1 GiB of memory. It is most appropriate for high-assurance environments where security needs outweigh infrastructure efficiency.

3.4. Tenable

Tenable is one more agentless security platform focused on visibility, compliance, and vulnerability management. It can also be integrated directly with the cloud provider's platform to assess configuration hygiene and identify only known risks. The agent also provides modes in which a deeper runtime insight is provided, and the platform allows optional lightweight agents to be deployed selectively. The agent has very modest requirements—about 200 MiB of memory and 100 millicuries of CPU—providing enhanced visibility without significantly affecting the overall application performance. Tenable’s architecture and scanning methodology make it an ideal candidate for organizations prioritising auditability and governance, even if runtime intervention is not their primary concern.

4. Comparative Analysis Security Agent

A detailed understanding of differences between container security agents will require a deeper analysis of not more than a glance at marketing claims. To develop our analysis, we draw out a specific requirement on application performance with meaningful insights into the security agents' minimum requirements. This paper assessed by comparing Red Hat ACS, Wiz, SentinelOne, and Tenable based on real-world criteria in the telecom application field that reflect how security tools are used in operational environments. Each tool is compared across detection techniques, vulnerability scan models, machine learning capabilities, cloud compatibility, and overall resource utilization impact under multiple application load conditions.

Red Hat ACS and SentinelOne primarily showed more compatibility with telecom workloads, where high reliability is a must. As both agents have a strong performance in runtime protection, both leverage deep and easy integration into Kubernetes via eBPF and kernel-level monitoring. Within OpenShift environments, the Red Hat ACS is very well suited, offering native admission control policies and Kubernetes-native deployment. Its ability to intercept and enforce policies at runtime makes it a preferred choice for telecom applications, which must protect data integrity at every acceptance.

Table 1. Feature Comparison of Container Security Tools

Feature	Red Hat ACS	Wiz	Sentinel One	Tenable
Detection Efficacy	High	Medium	High	Medium
Runtime Protection	Yes	No	Yes	Limited
Compliance Support	Strong	Strong	Medium	Strong
CI/CD Integration	Deep	Strong	Basic	Basic
Agentless Option	No	Yes	No	Yes
Resource Overhead	Moderate	Low	High	Low
Cloud Coverage	Kubernetes	Broad	Kubernetes Host	Kubernetes, ECS
Ease of Deployment	Moderate	High	Moderate	High

Wiz and Tenable are both light agentless deployment models, prioritizing cloud-wide visibility and compliance reporting over direct intervention. Wiz is widely accepted in multicloud platforms with both containers and VMs hosted and has direct integration into CSP metadata with configuration scanning capabilities. Wiz offers runtime protection; it excels in vulnerability management and is well-suited for applications that are deployed in a hybrid cloud infrastructure.

From the Use of machine learning standpoint, SentinelOne offers local support for various ML modes, which provides a potential advantage in detection capabilities. Sentinel One AI engine is configured to continuously learn from runtime behavior, allowing it to identify anomalies without relying on static signatures. The Wiz model uses ML for risk prioritization and sensitive data classification; this is done through its integration with models like LLaMA and NVIDIA's NIM framework. These ML features enhance the day-to-day operational teams with visibility and contextual risk understanding, particularly for cloud assets. CI/CD and the workflow pipeline integration with security agents are key in controlling the security risk with software deployments. Red Hat ACS provides a wide variety of tools and support, like Argo CD, OpenShift, and Jenkins, enabling engineering teams to embed security pre- and post-checks earlier in the development process. Wiz, which is much suited for multi-cloud platforms, supports integration with infrastructure-as-code pipelines like Terraform and GitHub Actions. At the same time, SentinelOne and Tenable provide some basic features on such platforms.

Overall, the model provides various options for multiple application types, and the choice to pick a specific platform comes down to organizational goals and operational needs. Agent-based solutions like Red Hat ACS and SentinelOne offer deeper protection and intrusion detection with runtime enforcement and behavioral analytics but consume more resources. Agentless tools like Wiz and Tenable deliver deeper metrics and insights into cloud visibility and enable faster deployment to various application platforms with a much lower impact on system performance. The table below provides very high-level details on key features offered by each platform and provides a guide for the engineering team to meet their organizational goals.

Table 2. Initial intrusion detection methods

Tool	Instrumentation Method(s)	Agent Type
Red Hat ACS	Daemon Set with kernel-level hooks, Admission controllers, and eBPF for runtime events	Agent-based
Wiz	API polling, CSP metadata analysis, Snapshot scanning	Agentless
Sentinel One	Daemon Set agent, Kernel hooks and syscall tracing, eBPF, Machine learning-based anomaly detection.	Agent-based (heavy)
Tenable	Cloud API scanning, Optional agent, Passive network traffic analysis, Container image scanning	Primarily agentless

Table 3. Machine learning model modes

Tool	ML Component	Use Case
Wiz	Risk Prioritization	Cloud metadata modeling
Wiz	LLaMA/NVIDIA NIM	Data classification
SentinelOne	Static AI Engine	File and binary classification
SentinelOne	Behavioral AI Engine	Runtime behavior and anomaly detection

5. Critical Security Vulnerability

5.1. 2022 – CVE-2022-0492

In early 2022, a critical vulnerability was highly visible in the Linux operating platform using unprivileged user namespaces. In Kubernetes clusters, the vulnerability was seen to have provided the attackers with the ability to escape the container boundary and escalate privileges on the host system, particularly when mandatory security requirements like AppArmor or second were not fully configured.

This incident served as a quick wake-up call to all cloud administrators, illustrating that even hardened systems could be exposed if kernel-level safeguards are overlooked. It drove the initiative in the organization to apply a security policy with more strict roles and access privileges and to have a feature to react and adapt to runtime threats.

5.2. 2023 – CVE-2023-3676

In 2023, Kubernetes base release's admission webhook handling found this critical high-severity vulnerability. The flaw provides loopholes where attackers can craft malicious requests; attackers could easily bypass cluster admission control policies and inject unauthorized workloads. Since webhooks are heavily used in cloud operations to enforce security and governance, the exploit exposed one critical weak point in Kubernetes control plane logic. The proactive fix for this CVE prompted many teams to re-evaluate their webhook configurations, applying zero-trust principles at every cluster policy endpoint.

Table 4. Critical vulnerabilities by year

Year	Number of CVEs Reported (Critical)	Notable Examples
2022	18	CVE-2022-0492, CVE-2022-23648
2023	24	CVE-2023-2431, CVE-2023-3676, CVE-2023-2728
2024	31	CVE-2024-10220, CVE-2024-29990, CVE-2024-9042

5.3. 2024 – CVE-2024-10220

This vulnerability was found in the deprecated gitRepo volume in a cluster. The vulnerability allowed attackers with gitRepo access to have pod-creation rights on the cluster to run unauthorized code by injecting malicious Git hooks into workload start-up routines. The issue was highly impactful where it was since up to Kubernetes version v1.30.2. The immediate mitigation step was to block CSI-based volumes and implement volume control methods to remove legacy volumes from local repositories. The flaw highlighted the most avoided work in every organization: cleaning up old records and volumes.

6. Experimental Setup

The Setup was deployed with a focus on having a detailed comparison of each security agent concerning the application under real-world operating conditions, and the deployment of two parallel Kubernetes clusters in telco enterprise-grade environments was keen in evaluating the Public vs Private offering of Kubernetes clusters in the market. The Amazon Web Services (AWS) with in-house Elastic Kubernetes Service (EKS) was publicly hosted, while the second was in a controlled lab with Red Hat OpenShift Container Platform (OCP). For the onboarded workload telecom applications, each platform was configured to use natively certified cloud storage solutions such as Amazon S3 and Elastic Block Store (EBS) for AWS and Ceph-backed OpenShift Data Foundation (ODF) for Red Hat.

Both clusters were deployed with similar workload application capacity configuration with 20 worker nodes, three masters, and three storage with CICD pipelines to support telecom-grade workloads for IMS services like IR92(voice), video IR(94), messaging, one-on-one and group chat, and conferencing. The application dimensioning was measured to support up to 1 million simulated users and reflect near production capacity and session volumes in a telecom environment.

The Lab setup used multiple production reflective busy hours trends to simulate the peak usage. Traffic was generated using multiple levels of user capacity load on applications, including a TAS (Telephonic Application Server): the SIP-based clients provided real-world UE sessions like registrations, mobile-originated and terminated calls, and multiparty conferences and real-time messaging. The simulation load was generated using the script from Spirent test equipment and SIP proxy simulation tools, with the call model taken directly from production busy hour sessions.

The test was conducted on multiple days, with each day, the user capacity increased by 50,000 users, allowing us to gradually build up traffic to a total of 1 million users while also capturing the gradual details on security agents. This approach enabled us to study how clusters and their associated security layers adapted to growing demand and diverse service interactions.

Security agents were deployed in both environments with flexible configurations, allowing us to turn Wiz, SentinelOne, Red Hat ACS, and Tenable on or off independently or in combination. Each tool was evaluated in 24- to 48-hour test windows during different traffic milestones, allowing us to compare their performance in stable and escalating load conditions.

The analysis captured a wide range of performance and security indicators. Beyond resource consumption (e.g., CPU, memory, and I/O), we also measured the ability of each agent to detect and respond to live threats. Controlled simulations included:

- Reverse shell intrusions
- DNS spoofing and response corruption
- Privilege escalation attempts
- Distributed Denial-of-Service (DDoS) attacks
- Malicious container behavior mimicking insider activity

This testbed provided a robust and practical framework for comparing agent capabilities. Unlike synthetic benchmarks, our Setup reflected real operational challenges, giving us a clear view of how well each solution performs in a demanding, cloud-native telecom environment under high user density and dynamic traffic patterns.

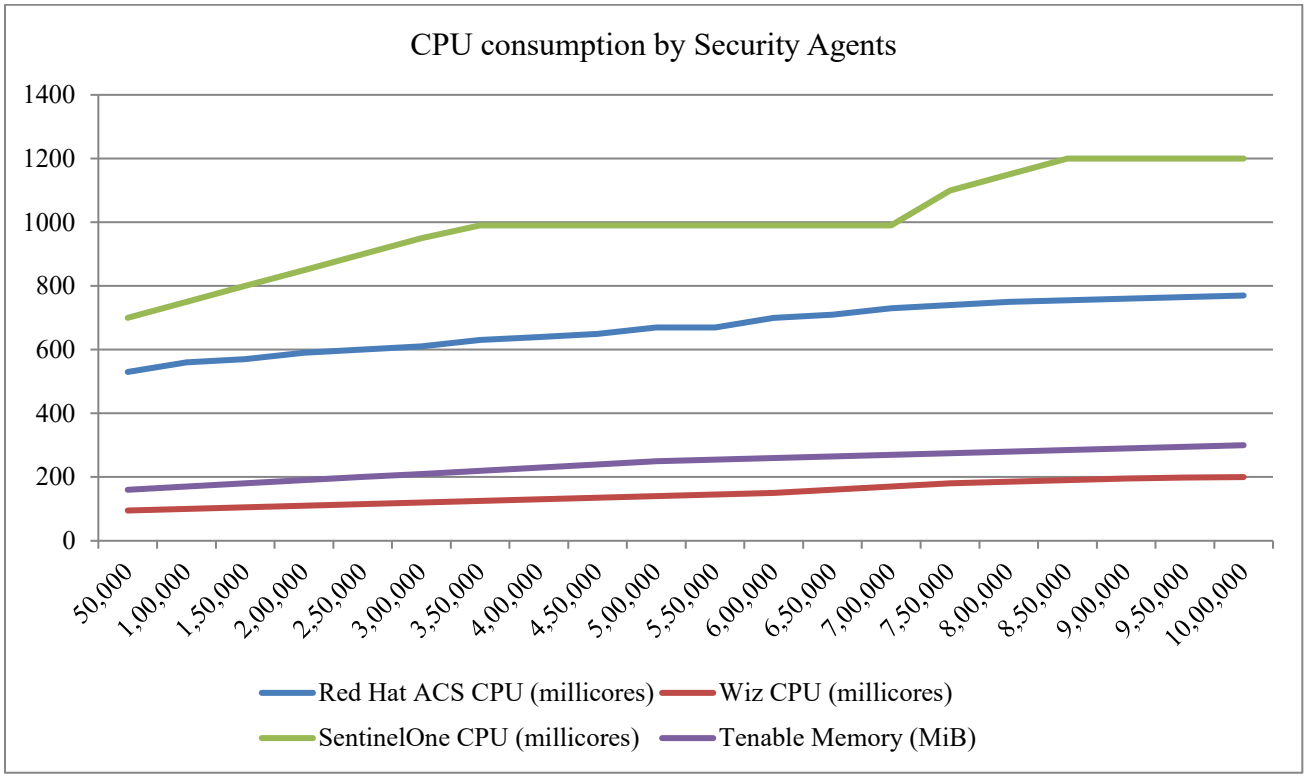

Fig. 1 CPU Consumption by Security Agents

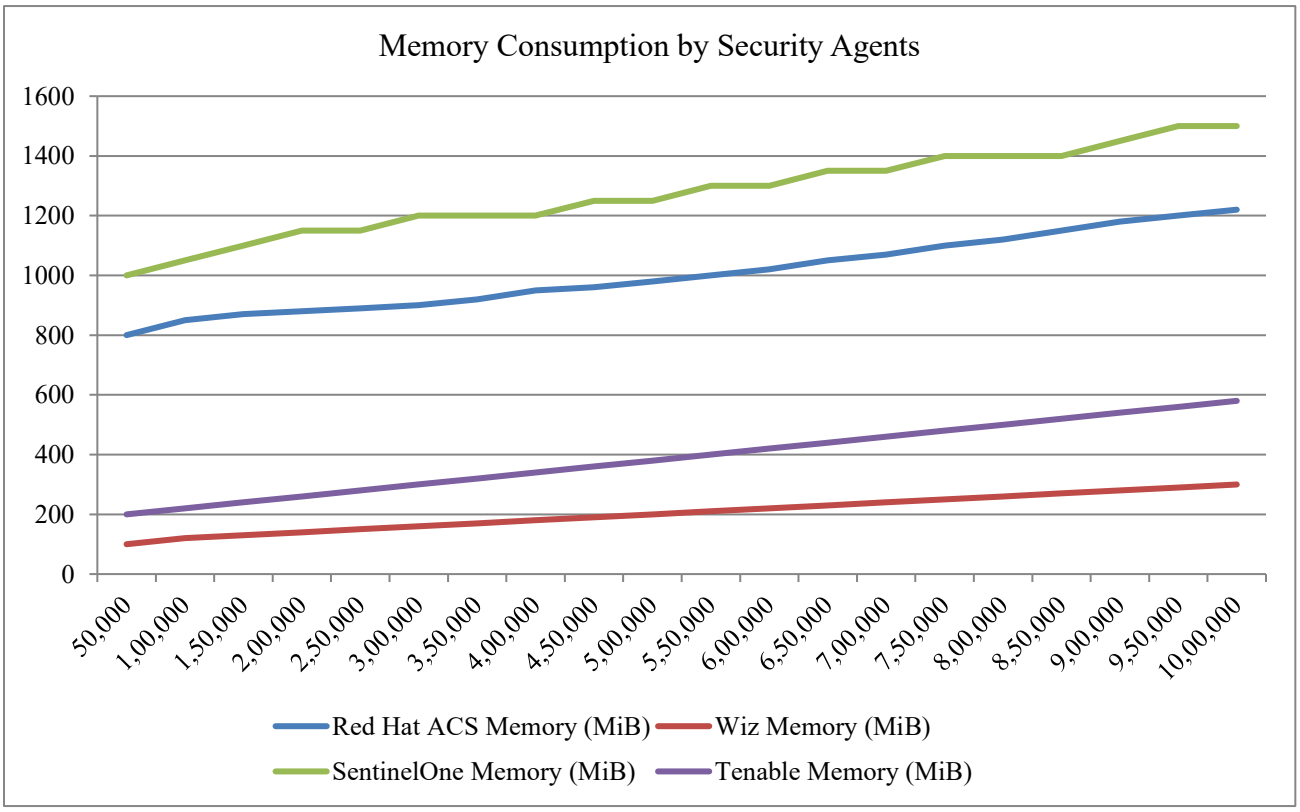

Fig. 2 Memory Consumption by Security Agents

7. Application Overhead with Security Agents

The performance impact of such agents depends on how each agent is architected within the cluster, how it operates, and the frequency at which it monitors system activities with proactive measures to tackle them in case of vulnerability detection. The typical agent-based solutions, like Red Hat ACS and SentinelOne, are configured on the cluster as DaemonSets or sidecar containers across each worker node. These agents provide multiple features for robust, real-time detection and protection by continuously analyzing the management network on workloads for any runtime behavior changes and intercepting threats as they emerge. The features and services always come with a cost on infrastructure. The sidecar agents have dedicated requirements on CPU and Memory and tend to overcommit their reservations in case of reactive threat mitigations, which can strain clusters running dense workloads.

The lab environment uses a highly resource-constrained environment, with reserved capacity for the security agents, and any overcommit could lead to slower performance on the application pod by lowering node efficiency and, in some cases, restarting the complete worker instance. SentinelOne is a particular agent with a very high resource-intensive requirement due to its features providing behavioral AI and continuous system tracing. The other two agents, agentless platforms like Wiz and Tenable, have less impact on the cloud platform and use a very less intrusive approach. These tools do not reside inside the cluster. Instead, they connect externally via cloud provider APIs to assess configurations, scan metadata, and detect risks across cloud assets. This design keeps resource usage minimal within the cluster itself. While this approach reduces the performance impact on workloads, it also limits the agent's ability to detect or react to threats in real-time, particularly those that arise inside running containers.

In the experimental clusters, these theoretical trade-offs were visible. During instruction-level testing—such as privilege escalation attempts and reverse shell triggers—agent-based tools exhibited sudden CPU and memory spikes**,** directly impacting application capacity. In some instances, service pods were delayed or rescheduled due to resource contention with the security agent itself. This was especially evident when reaching higher user concurrency levels, where any overhead was amplified. Additionally, some agents, particularly those with behavioral or forensic logging features, demanded significant storage bandwidth to write telemetry and event logs. This was seen in the event logs on both AWS (S3/EBS) and Red Hat (Ceph ODF) clusters. When the application user capacity peaked at 1 Million users, these events on storage I/O contention led to latency SIP session binding into the local storage platform and call data record processing. However, this slowed the cloud application storage solution, affecting the security agents' delayed detection of threats caused by lagged data ingestion.

To keep a healthy balance between application and security agents, which need to coexist to have better protection and overall performance, the engineering and security teams must adopt well-planned resource management considering the rainy cases where cluster capacity is over-exhausted. The runtime agent with right guard rails and dedicated CPU pinning and memory limits would leverage Kubernetes to schedule these agents with higher Quality of Service (quality of service) tiers and isolate agent processing with application workloads on dedicated CPU-pinned zones, thereby helping to prevent resource contention. Furthermore, the data restoration, fine-tuning log retention, and the storage I/O max rate with multiple verbosity levels can minimize the performance degradation caused by agents that depend on data for ML algorithms to prevent runtime operations. These observations can be seen on the chat below, showing each agent's performance at two application capacity variations with high call volume and heavy read-write telecom workload instances.

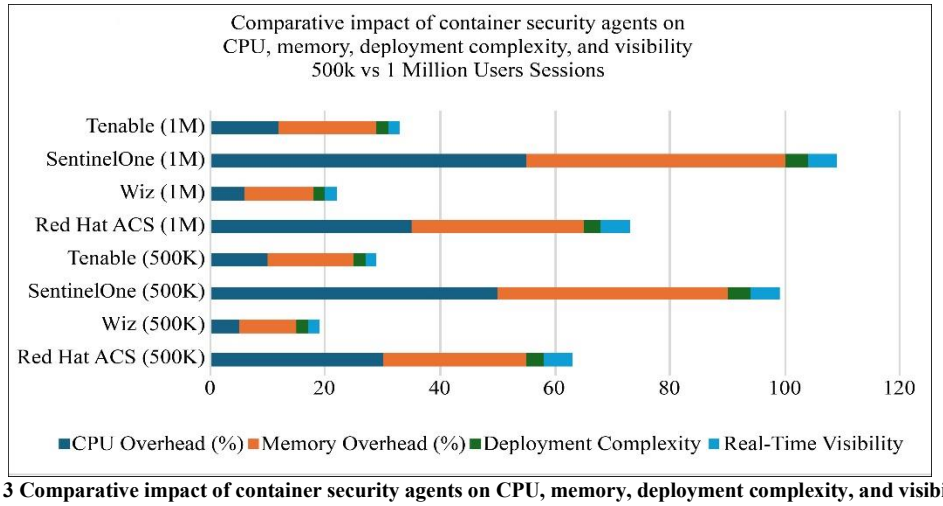

Fig. 3 Comparative impact of container security agents on CPU, memory, deployment complexity, and visibility

Table 5. Pods Deployed on Node

Pod Name	Type	vCPU millicuries	Memory
app-voice-1	Application Pod	1000	2048 MiB
app-video-1	Application Pod	1500	3072 MiB
app-messaging-1	Application Pod	800	1536 MiB
app-conference-1	Application Pod	1200	2048 MiB
security-agent-s1	Security Agent (S1)	1000	1024 MiB
kube-proxy	System Daemon	200	256 MiB
kubelet + system	System Services	300	512 MiB

8. Underlying Risk with Security Agents

Example: Resource Allocation on a Single Kubernetes Worker Node

8.1. One Cluster Node Specs

- vCPUs: 8 (8000 millicuries)
- Memory: 16 GiB (16384 MiB)

Total Resource Usage:

- Total CPU Requested**:**
  1000 + 1500 + 800 + 1200 + 1000 + 200 + 300 = 6000 millicuries (75%)
- Total Memory Requested**:**
  2048 + 3072 + 1536 + 2048 + 1024 + 256 + 512 = 11,496 MiB (≈70%)

8.2. Intrusion Event Scenario

When the security agent detects malicious activity—e.g., a reverse shell attempt **or** privilege escalation—it triggers intensive logging, behavior tracing, and memory analysis. This led the security agent to overcommit the CPU requirements to perform the operation tied to analyzing and reporting. In our testing, we saw CPU overcommits exceeding 200% in some cases for Sentinel One and RedHat ACS security agents.

While container security agents are critical for defending modern workloads, their Use introduces a unique set of operational risks that, if overlooked, can compromise the systems they aim to protect. These risks do not negate the need for such tools but highlight the importance of deploying and managing them thoughtfully.

One of the most common challenges is resource overcommitment**.** Agent-based security agents primarily include Red Hat ACS and SentinelOne, which run as DaemonSets or sidecar containers. Under routine sunny day conditions, their impact may be minimal. However, when the system is under attack or has an anomaly pattern detected with system access, these agents without strict guardrails have been seen to have more than the reserved consumption on CPU or memory.

If resource limits are not properly set, this behavior can put stress on the application pods hosted on a worker node, potentially leading to pod evictions in case of resource over-commitment, thereby causing application performance degradation or creating signalling lags for mission-critical services, especially in clusters where applications are tightly coupled with very less resource headroom.

Another major concern with such a security agent involves privileged access exposure. To provide deep visibility and runtime control from vulnerability threats, ACS and SentinelOne require elevated permissions to take preventive steps, such as access to cluster networking, application container runtime sockets, and ACL rule updates. If any of these permissions are misaligned or misconfigured, they can be exploited, they can become the entry points for attackers.

Application Latency overhead is a critical case for consideration, particularly in a cluster where application uptime is needed at 99.999 % at any given time, and performance is closely monitored or guaranteed through service-level agreements (SLAs) with customers. Security agents monitoring the POD-to-POD syscall access logs or performing kernel-level deep packet inspection can introduce minor, measurable delays in applications with user traffic sharing the same kernel space. Such delays are often acceptable in general workloads. However, in high-reliability workloads with application pods with affinity and anti-affinity rules, the impact is real-time applications' performance degradation with a minor spike in agents' drift in policy.

The other issue, which was proactively seen and captured in our experiments, is that deployments of these agents across all workload instances can sometimes drift and have inconsistent updates. In clusters deployed across multiple racks with hundreds of nodes, it is easy for agent versions to fall out of sync, either due to overlooked updates or custom configurations. The outdated agents in day-to-day operations are often not keenly monitored. They can be an easy loophole or security blind spots, reduce detection

accuracy, or cause compatibility issues when upgrades are planned.

Lastly, the key observation seen in testing was many false positives and alert fatigue, which were seen persistently from day 1. The agents that have trends to detect behavioural changes in access patterns are often seen to flag legitimate activity as malicious. This can lead to unnecessary operational overhead on organizations in teams working constantly to filter out these alerts from legitimate events. False events can also disrupt deployment pipelines. If automated responses are enabled in CI/CD pipelines, they can lead to terminating a healthy workload instance.

To mitigate these risks, application design, security, and operational teams must plan to implement best practices: with stringent resource boundaries, apply role-based access controls and prevent the need for privileged access by policies, standardize CI/CD workflows pipeline with scanning software images, and ensure alerting pipelines are integrated with SIEM or SOAR platforms for contextual triage.

The planning of these standards needs to be in every cloud-native application design; with these practices in place, security agents are highly trusted to provide deep protection without compromising application performance.

Table 6. Resource and Privilege Risk by Agent

Agent	Can Overcommit CPU	Needs Elevated Privileges	Notes
Red Hat ACS	Yes	Yes	DaemonSet with eBPF; enforce limits
Wiz	No	No	Agentless; low risk from deployment
SentinelOne	Yes	Yes	Runtime ML agent; high overhead risk
Tenable	No	No/Optional	Agentless or optional lightweight sensor

Properly planning cluster resourcing with dedicated quotas to security agents with the proper privilege controls and constant telemetry data monitoring is important to mitigate these risks. All agents must be tested fully in lab-stage environments before production rollouts.

9. Security by Design in Cloud-Native Infrastructure

Security is necessary for every application design and cannot be treated as an afterthought or something that can only be deleted when an incident is observed. With the growing cloud infrastructure and application adaptation to such environments, whether it is in public, private, or multi-tenant cloud infrastructure, security must be a part of the application deployment design. As organizations adopt and rapidly transform towards containerized architectures, the security loopholes and vulnerabilities to attacks also increase rapidly. Risks such as misconfigurations, allowing privilege escalation, and compliance expectations on drift become more likely and harder to correct if not implemented during initial deployment.

Cloud-native applications provide flexibility to be dynamic by nature. The platform provides applications with dynamic capacity by resource or instance scaling, span across multiple availability zones, and run as sidecars, which are very short-lived, interdependent workloads. The need to close any design gaps around the security in such fast-moving environments will need to be planned at the very beginning, during architecture planning and requirements gathering, and not patched in after the system goes live.

The key plan for such a cloud application design is to put the security requirements first and build the application around them security requirements. Teams should also determine what level of action needs to be taken during a security breach event by carefully identifying the most secure data and providing additional security controls in accessing such information, which compliance standards apply (e.g., PCI-DSS, HIPAA, NIST) and what kinds of threat models are best fit for the workload. Cloud deployment core architectural decisions should reflect these requirements, with zero trust, role-based privilege policy, network segmentation, and data encryption in flight and at rest.

The Cloud Security Principle includes the following:

- Continuous vulnerability and configuration scanning are used to detect and correct issues before they become exploitable.
- Identity and access governance**,** using RBAC and centralized IAM to control who can access workloads, data, and infrastructure.
- Runtime monitoring to catch behavioral anomalies, insider threats, and active intrusions.
- Automated compliance validation to generate audit-ready reports and reduce the burden of regulatory assessments.

To further improve operational resilience, especially in latency-sensitive environments like telecom networks, it is recommended that security agents have access to dedicated storage volumes**.** These volumes should have a defined path for mounting critical data events like logs, alerts, or forensic data without interfering with the primary application's I/O

in the storage cluster. This approach to storage design reduces the risk of write latency during high-traffic security incidents, such as network anomalies, DDoS attempts, or privilege escalations. It ensures that security telemetry does not interfere with or degrade application performance. For telecom-grade applications in particular, such resilience is crucial to maintaining uptime and user experience at any given time.

10. Budgeting for Secure Application Delivery in Modern Cloud Environments

The budget allocation reflects the organisation's goals in protecting and preventing security threats to mission-critical applications. In every project or application design, a substantial share of investment is always directed toward traditional development, deployment, and operational maintenance. The budget must prioritize security from the beginning and acknowledge its critical role in today's dynamic cloud-native application environments.

Findings from the comparative analysis of security agents with respective vulnerabilities observed on the platform support this argument of having the proper budget to plan, test, and implement security measures. The network breaching testing with and without security agents provided different setups, where in deployment, security agents observed and alerted the threat. In contrast, the cluster with the agents in monitor or inactive mode allowed back actors to access valuable data on the platform. When the agents are deployed without the right resource isolation or tuning, they can instantly overcommit, and this is primarily seen with SentialOne agents affecting the performance of core application workloads, particularly during traffic surges or threat detection events. The productive analysis from this test result shows the importance of having the right resource planning and budgeting the cost of such resource requirements at the beginning, which is essential for reliable and resilient application deployments.

The inclusion of the CI/CD pipeline in our deployment had a key role in enforcing a secure check on software, even before the application development lifecycle was started. In the test environment, any new software code or container image submitted to the platform is automatically subjected to vulnerability scanning before it can be promoted for deployment. Avoiding this step allowed critical vulnerabilities to be implemented on the cluster, which is prone to threats and attacks. Including a security scan at the CI/CD image push step, powered by the integrated security agents, ensures that only verified, compliant artefacts enter the runtime environment. By following this requirement, deployment pipelines help ensure software vulnerabilities and mitigation steps are addressed first, and by doing so, security becomes a seamless part of the deployment process, reducing the risk of introducing known vulnerabilities into production systems.

In summary, the planning and budget structure for product or application development that uses public or private cloud solutions supports application delivery's core objectives. It aligns with modern security practices' operational demands. The proper budget enables organizations to have security resources to deploy faster, respond to threats more effectively, and maintain compliance, all without compromising performance or scalability.

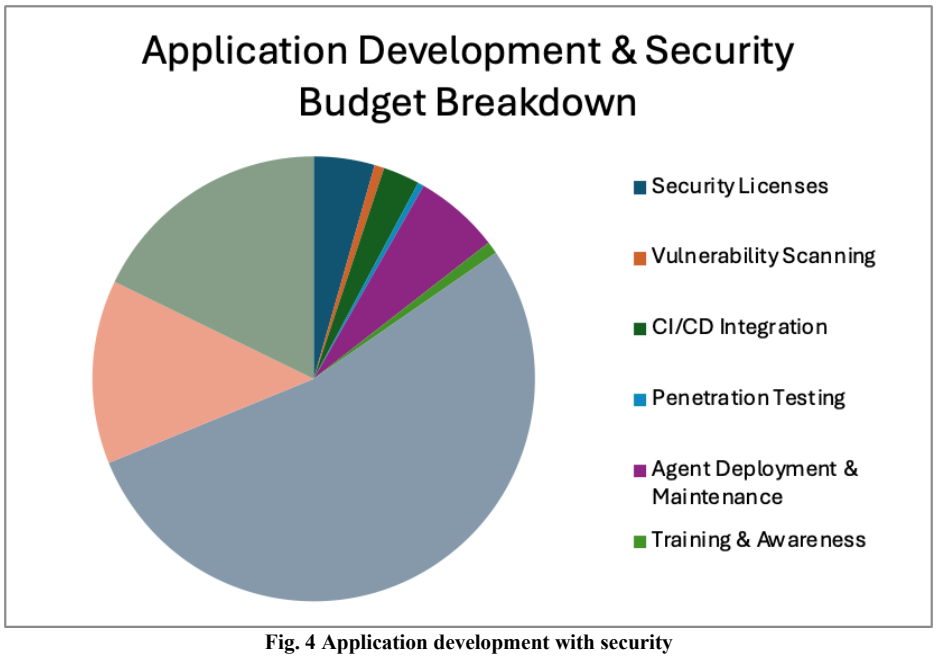

Fig. 4 Application development with security

11. Real-World Intrusion Prevention Success

As cyber threats grow more sophisticated, government agencies and public-sector organizations have responded by investing in smarter, more proactive security frameworks. In 2024, UK government institutions reported impressive results, collectively stopping over 15 million cyberattacks. The Met Office intercepted over five million phishing attempts alone, while the DVLA successfully blocked upwards of seven million distributed denial-of-service (DDoS) attacks. These outcomes were achieved by implementing modern security technologies that blend real-time monitoring, behavioral analysis, and dynamic runtime protection. These tools have become vital in shielding critical infrastructure from increasingly coordinated and targeted attacks.

Similar efforts have bolstered federal and state-level cyber resilience in the United States. The Department of Homeland Security's EINSTEIN system, designed to protect federal networks, now processes over 30,000 daily alerts. Meanwhile, state and local governments mainly depend on data from the MS-ISAC Albert system, the American nationwide network of intrusion detection sensors, to monitor and analyze suspicious activity across their digital ecosystems. These public systems have played a key role in helping software developers gain awareness of new threats and security loopholes, thereby preventing ransomware incidents, stopping unauthorized lateral movement, and identifying supply chain threats before they escalate.

Overall, in both academic research and industry findings, the public research data on cloud-native platforms have shown that runtime-aware tools, such as Red Hat Advanced Cluster Security (ACS) and SentinelOne, deliver significant advantages over traditional scanning solutions. Using dedicated monitoring agents like eBPF on kernel space and machine learning models to follow and detect subtle, behaviour-based threats in real-time, capabilities that signature-based tools typically miss. Public data domains that track and provide high critical vulnerabilities have consistently supported this claim, showing higher integration in sharing this information and visibility in proactive or reactive measures to control or avoid such incidents. The study provides good data points for organizations managing microservices or large-scale containerized applications, as security agents are rapidly becoming an essential part of a modern cybersecurity stack.

11.1. Reactive Agent Comparison (Red Hat ACS vs Sentinal One)

This section focuses on finding the advantages of ACS and SentialOne with respective reactive security mechanisms, which have grown to have a higher demand in telecom cloud application design. This is essential in the case of public-facing containerized application environments where threats can evolve and propagate quickly. These agents were tested for the following four different capabilities concerning the real-time response capabilities of four major container security platforms.

Table 7. Runtime reactive capability comparison

Tool	Real-Time Detection	Automated Response	Reactive Flexibility
Red Hat ACS	Yes (eBPF + Admission Hooks)	Yes (Policy Enforcement)	High
SentinelOne	Yes (eBPF + ML)	Yes (Kill, Quarantine, Rollback)	Very High

In the system privilege escalation task test, the data shows that Red Hat Advanced Cluster Security (ACS) and SentinelOne stood out for their strong runtime responsiveness. SentinelOne, with a more dedicated CPU and Memory distribution model, leverages behavioral analytics and machine learning to detect real-time anomalies. The reactive response to a threat by isolating compromised workloads was also seen to help prevent attacks and rollback changes. Red Hat ACS, meanwhile, with a full policy control-based approach, took control of such an event by creating a new webhook policy update for runtime instrumentation.

The low-cost agentless platforms like Wiz and Tenable are seen to show more alerts and data analytics on posture visibility and misconfiguration detection. The agents were ineffective in a live attack when system access privilege requests were granted to a service account. These agents have robust scanning and compliance capabilities, and the agentless architectures are fully equipped to handle continuous system data monitoring for a live attack. This makes them less suited for real-time threat prevention, especially in cases where immediate response is required to maintain data integrity.

In handling data integrity under a security breach, the lab tests show that reactive security agents are particularly valuable in such environments. In this case, we tested our messaging application platform where user data is encrypted. However, the archived data is stored on cloud storage platforms, and any downtime of this data or application handling these events has direct business or operational consequences. The well-known public applications where data security is highly valued are financial services**,** healthcare systems**,** and e-commerce platforms**.** In such use cases, having a dedicated and continuous monitoring security framework is a key requirement for application reliability. Both Red Hat ACS

and SentialOne in our analysis have given positive observations in alerting and preventing zero-day threats, lateral movement, insider attacks, and policy violation use cases.

The Red Hat ACS platform More Us was found to have a more suited approach to applications that provide quick features and have a short life span on given containers**,** with real-time APIs**,** or have fully automated CI/CD pipelines**.** The Red Hat ACS, compared with SentinelOne, has lower budgeting requirements and provides real-time detection capabilities, making it idle for fast-changing environments.

12. Conclusion

The data-driven analytical study with a group of voice, video, and messaging applications shows that security services and policies are not a universal one-size-fits-all solution for securing containerized environments. Each platform assessed in this study—Red Hat ACS, Wiz, SentinelOne, and Tenable—offers distinct advantages tailored to different operational needs and deployment priorities. The test data points to real-world user traffic patterns at various system load levels, showcasing the importance of aligning computing resources with specific needs to meet the organization's commitment to providing secure, reliable applications.

Solutions such as Red Hat ACS and SentinelOne are well-suited for scenarios in telecom applications where runtime visibility and rapid threat response are critical, especially in voice services. Their agent-based architectures provide fine-grained control and behavioral analysis options. They are particularly valuable in regulated sectors or environments where workload integrity must be rigorously enforced to avoid application downtime. These tools empower operational and security teams to monitor and intervene in real-time, helping mitigate the impact of live threats before they escalate.

Conversely, Wiz and Tenable provide a lighter footprint and greater ease of deployment in multi-cloud tenant platforms. The agentless models allowed a quick establishment of security baselines across complex, multi-cloud, multi-zone, restricted network cloud environments without reengineering the underlying infrastructure. These agents have been more stable, consistently perform, and are well-suited for teams focused on rapid scalability, compliance tracking, and minimal operational overhead.

Wiz security agent, among others, has received much internal support from our development teams as it has a good framework to integrate seamlessly into DevSecOps workflows. As many of the IT applications are light and have little dependence on the underlying hardware platform, the agentless security platforms were keen to provide security measures to developers. Wiz enables risk prioritization and policy mapping without the complexities of managing in-cluster agents. This approach has proven effective for organizations seeking agility and actionable security insights.

Ultimately, the decision to adopt a particular platform should be guided by an organization's security teams, with application and operation teams in part of the risk tolerance, architectural landscape, and long-term security objectives requirement planning. A thoughtful evaluation of these trade-offs ensures that the chosen solution meets today’s needs and scales effectively with future operational growth.

References

[1] Kubernetes, Production-Grade Container Orchestration. [Online]. Available: https://kubernetes.io/
[2] Red Hat, Advanced Cluster Security for Kubernetes. [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift/advanced-cluster-security-kubernetes
[3] Wiz, Protect Everything you Build and Run in the Cloud. [Online]. Available: https://www.wiz.io/
[4] SentinelOne: Autonomous Endpoint Protection, Sentinelone. [Online]. Available: https://www.sentinelone.com/resources/sentinelone-autonomous-endpoint-protection/
[5] Tenable. [Online]. Available: https://www.tenable.com/
[6] The Sun. [Online]. Available: https://www.thesun.co.uk/tech/34784827/number-cyber-attacks-met-office-revealed/
[7] U.S. DHS, Einstein Intrusion Detection System. [Online]. Available: https://en.wikipedia.org/wiki/Einstein_(US-CERT_program)
[8] Center for Internet Security, MS-ISAC Albert Network Monitoring. [Online]. Available: https://www.cisecurity.org/ms-isac
[9] MITRE, ATT&CK® Matrix for Containers. [Online]. Available: https://attack.mitre.org/matrices/enterprise/containers/
[10] Cloud Native Computing Foundation (CNCF), Cloud Native Security Whitepaper. [Online]. Available: https://www.cncf.io/reports/cloud-native-security-whitepaper/
[11] CSE-CIC-IDS2018 Dataset, Canadian Institute for Cybersecurity. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html
[12] CIC-IDS2017 Dataset, Canadian Institute for Cybersecurity. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html
[13] National Vulnerability Database, NIST. [Online]. Available: https://nvd.nist.gov/