International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group | Volume 3, Issue 3, 148-158, 2022
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V3I3P115

Original Article

Kube Agent Hardening for Fleet-Wide Secure Telemetry

Rohit Reddy Gaddam[1], Kalyan Krishna[2]
[1]Sr. Site Reliability Engineer.
[2]Infrastructure Engineer.

Abstract - *Kubernetes has become the technological pillar of the cloud and, more specifically, the cloud-native infrastructures, that is, the infrastructures that are implemented by means of cloud services. It has thus facilitated all those organizations that wanted to manage all their containerized workloads through wide and distributed environments. Kube agents are at the core of this ecosystem they are the smallest, lightweight components that collect and send telemetry data, which is necessary data if we want to be able to monitor observability and performance and keep the clusters in compliance. We briefly refer to the problems of non-uniform agent configurations, the dangers of agent communication channels, as well as the issue of operators that have to work really hard just in order to distribute security controls equally throughout all kinds of clusters. For solving those problems, we come up with a plan that includes a combination of zero-trust principles, the procedure of rotating certificates throughout the whole fleet and the utilization of fine-grained policies that are related to roles and, at the same time, are completely integrated with Kubernetes-native constructs. According to our research, securing telemetry is not only about encrypting the data that are being sent but it is also about being able to develop confidence against the situations of the agents not being properly configured, of the insiders who may pose threats and also of the utilization of compounding persistence-type malware, which may leverage agent-level weaknesses.*

Keywords - *Kubernetes Security, Kube Agent Hardening, Secure Telemetry, Zero Trust, Container Security, RBAC, Fleet Management, Observability, Threat Detection, Cloud-Native Security.*

1. Introduction

1.1. Challenges in Kubernetes Telemetry

As the organizations decide to use Kubernetes to manage their containerized projects, telemetry has become one of the main things that show the visibility of operations. Telemetry agents whether they are Prometheus exporters, Fluentd log forwarders, Open Telemetry collectors, or custom sidecars provide the raw data necessary to understand system performance, detect anomalies, and maintain compliance. Even so, the attributes that essentially make Kubernetes as powerful as it isi.e., its dynamism, scalability, and the ability to abstract infrastructure complexity have a side effect of impeding secure and reliable telemetry collection significantly.

One of the biggest hurdles is related to dynamic and ephemeral workloads. Containers might have a lifetime of a few minutes or might live only for a few seconds; thus, they can be created and deleted as the demand goes up and down. As a result of being so transient, telemetry agents have to perform all of their operations, such as discovering, attaching, and releasing data sources, continuously, but at the same time, they have to guarantee the security and fidelity of the data. Fixed expectations or assumptions about node identity, network topology, or lifecycles of the workload are no longer applicable in such surroundings; hence, securing telemetry pipelines has become inherently more complex.

Another major problem is the complexity of multi-cluster observability. In such a scenario, an enterprise may not have one but rather several clusters of Kubernetes (sometimes, even hundreds of them), which may be spread across not only multiple cloud providers but multiple geographic regions and edge locations too. Each cluster would have its own monitoring stack, its own set of policies, and its unique patterns of integration. To gain a consolidated and credible telemetry phyletic over such a vast area is not a walk in the park. It necessitates agent configuration and policy enforcement that is consistent throughout the fleet.

However, equally important is the fact that telemetry has a dual nature. It is on one hand just a necessary visibility enabler that allows security daemons to have access to data streams for incident detection, compliance audit, and hunting of threats. While on the other hand, the telemetry pipelines are the most potential source of the attackers' intentions. Agents usually require elevated permissions to collect and forward data, and they also are in constant outbound communication. Thus, they are in a perfect position to be exploited either by privilege escalation, lateral movement, or data exfiltration. Telemetry agents may unintentionally leak the metadata of workloads, configurations, or even user interactions, which gives the attackers free access to the reconnaissance material they need.

At the end of the day, we are still dealing with security gaps in the communication line of agents. A lot of telemetry programs are built upon insecure defaults, for example, unencrypted HTTP transport, very broad cluster-wide privileges, or static credentials without the rotation. This, in turn, might give rise to situations where telemetry data is intercepted, tampered with, or even taken by unauthorized endpoints. Privilege escalations, i.e., occurrences when agents that have been compromised acquire access to workloads or host nodes that they are not intended to, thus magnify these dangers. As a whole, these gaps turn telemetry from a basic enabler of operations into a latent risk when deployed at scale without hardening.

1.2. Problem Statement

Telemetry is essential for K8s systems; however, there are no uniform standards in place to guide the classes on how to cool down telemetry agents Moreover, organizations often allocate significant budgets for securing applications and network perimeters, and yet the components carrying telemetry are mostly ignored or treated as the operating team's afterthoughts. For instance, Prometheus exporters might be unprotected and leak secrets, while Fluentd or Logstash forwarders may transmit logs in clear text. OpenTelemetry collectors could be overprivileged. The security of custom sidecars that came suddenly by reason of observability gaps is questionable because they might not even have the most basic types of input and safe programming practice.

What is more, there is a risk of hacked agents being used to carry out attacks on different fleets which is very scary. Agents are usually deeply trusted and therefore have access to secure areas and data. So an attacker who successfully hacks one agent can take control of the affected node, and probably move over to other nodes plus gain control over the area where supervisors work. Environments of multi-clusters also exaggerate these dangers.

Besides, the problem becomes even bigger because of security disunited policies that exist between clusters and regions. Enterprises receiving different kinds of managed k8s services or disbursing cluster administrations into distinct business units tend to have the most significant difficulties when it comes to establishing uniform telemetry controls. Simply speaking, Kubernetes telemetry is currently fractured, delicate, and openly accessible. Organizations are still open to both random and targeted attacks that focus on the observability pipeline without having a universally acceptable method of hardening telemetry agents.

1.3. Motivation

The reasons for closing these gaps in telemetry security stem from both external influences and internal necessities. Externally, among various drivers, regulation and compliance requirements lead organizations to implement strong controls on the collection, transmission, and storage of operational data. For instance, the General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), and Payment Card Industry Data Security Standard (PCI DSS) set up a framework requiring that sensitive data, including logs, traces, and metrics that may contain user identifiers, transaction details, or protected health information, have strict protection. If security measures are not implemented properly, there is a possibility of telemetry streams being the source of a breach; in that case, the organization will also be liable for regulatory penalties in addition to losing the trust of their customers, and, therefore, their reputation will be damaged.

From the internal perspective, getting to know the performance of agents in the organization through the negative examples of breaches caused by the misconfiguration of agents not only stresses the urgency but also the necessity of the issue. The main problem in almost all the incidents leading to telemetry endpoint security weaknesses is that the endpoints have been left open, thus allowing the attackers to retrieve the needed information to penetrate deeper into the system without being detected. Consequently, these breaches are a lesson that telemetry, which was once ignored as a passive subsystem, is now becoming an active target for exploitation.

Moreover, the implementation of zero-trust concepts will, in addition, lead the overall work of telemetry pipelines to be more secure. Under a zero-trust model, the architecture of trust does not allow for any component, whether it be an application, a service, or an agent, to be at any time fully trusted. In essence, they need to verify, get authorized, and even encrypt their communications every single time. Similarly, telemetry is putting these concepts into practice which means any data being transferred between the agents and collectors needs to be encrypted using mutual TLS, privileges should be as low as possible, and there should be a record of all data flows.

One more thing is that there is a practical necessity wrapped up in it as well which is to make sure that telemetry data are correct for the incident response and threat hunting. Security teams rely heavily on telemetry to rebuild attack timelines, figure out which assets have been compromised, and at the same time validate the steps of remediation. The process of incident response gets hampered if the telemetry data, that is, if it is counterfeit, incomplete, or falsified. In this regard, the well-secured kube agents play the role of protectors and at the same time they guarantee the integrity of data, so that instrumentation can still function as a source of truth in the forensic investigations sector.

2. Literature Review

2.1. Kubernetes Native Security Mechanisms

Kubernetes has developed and is now the de facto standard to manage container orchestration, while the security model it offers is an excellent base to protect the workloads. The Role-Based Access Control (RBAC), which is at the center, defines the authorizations of the users, service accounts, and system components. The RBAC allows close access management, thus giving the administrators the possibility to limit telemetry agents just to those privileges that are necessary for grabbing and exporting the data.

However, the number of misconfigurations is still high despite RBAC's versatility. One of the main problems caused by too many permissive roles is privilege escalation paths. Several studies (e.g., Red Hat reports on Kubernetes security posture) are constantly pointing out that giving service accounts too many privileges is the most prevailing misconfiguration in production clusters.

The Role-Based Access Control (RBAC) system was supported by Pod Security Policies (PSPs), which are now being replaced by Pod Security Admission (PSA) and policy managers like Open Policy Agent (OPA) and its Kubernetes-native variant, Gatekeeper. These devices impose limits on workloads, e.g., not allowing containers to run as root or share host namespaces. Although PSPs and OPA can easily identify least-privilege configurations, their direct relation to telemetry agents is not always clear. Agents are often given raised capabilities (e.g., host filesystem or kernel access for node metrics), and so policy exceptions that lower security levels of the entire cluster are made. NetworkPolicies is another critical security feature that governs traffic flow between pods, namespaces, and external endpoints. NetworkPolicies for telemetry can, for example, be used to allow agents to communicate only with arbitrarily selected collectors or gateways.

2.2. Existing Telemetry Solutions and Their Vulnerabilities

Various telemetry tools have become an essential part of Kubernetes environments. They are mutually exclusive in terms of features and weaknesses.

Prometheus is one of the most popular monitoring systems that is built around a pull-based metrics scraping model. Nevertheless, as it stands now, the default Prometheus exporters frequently reveal endpoints with no authentication or encryption, thereby being easily accessible by unauthorized users. Some CVEs have been documented outlining the security loopholes, e.g., metrics endpoints revealing sensitive node configuration, resource utilization, or service topology details.

Fluentd and Fluent Bit are well-known log collectors and forwarders of logs to central systems. Their small size and efficient design are perfect matches for Kubernetes sidecar patterns; however, insecure default settings are still prevalent. Instances of misconfigured Fluentd have been reported where logs were transferred via unencrypted TCP hence sensitive application logs can be easily intercepted by cyber attackers. Furthermore, the nature of the Fluentd plugins mostly third-party contributions makes security breaches more likely.

Open Telemetry is a new standard that encompasses all the metrics, logs, and traces. The project is very ambitious, but it is at different developmental stages for different ones of its components. In most cases, Open Telemetry collectors have very extensive permissions, and some of the setups require that agents be given privileged access to the kernel to get the data they need. The researchers also pointed out that pipeline insecurity could be a source of data exfiltration if no proper authentication and encryption are enforced during the transition.

One of the most recurring things that can be spotted within all these instruments is the issue of insecure default values. The deployment of agents from Helm charts or manifests is usually done with the main goal of making the deployment as easy as possible, security is somehow sidelined. Encryption may be seen as an optional rather than an indispensable feature, authentication may be turned off to make the scraping easier, or RBAC roles may be created with cluster-admin privileges. All these shortcomings serve as a reminder that the issue of security should be front and center in the telemetry agent deployment.

2.3. Research on Agent Security in Distributed Systems

The security threats raised by agents in distributed systems have been well known in the academic literature for a long time. Such autonomous components energize the system, but at the same time, they create a new security perimeter, which is difficult to protect. The study on mobile agents in the late 1990s focuses on agent impersonation, malicious hosts, and data integrity as the main threats. Despite many changes in cloud orchestration paradigms, these concerns are still relevant.

The dual nature of agents as essential data sources and high-value targets is stressed in recent research on cloud observability and distributed tracing. The paper published in 2020 in IEEE Transactions on Cloud Computing informed that monitoring agents in distributed cloud systems might be turned over to release sensitive information or to provide backdoor access. It should not be surprising that these messages are in line with Kubernetes situations where telemetry agents are frequently working on the other side of the trust boundary (for example, between nodes, namespaces, or clusters).

Also, concern in papers for the industry is similar. Google's Site Reliability Engineering (SRE) methodology is an example; it supports the pipeline for monitoring, which leads to incident response. Besides that, CNCF community discussions (2021–2023), which talk about observability agents, are a character that keeps coming up as the weakest link if left unsecured.

2.4. Case Studies: Telemetry Attacks and MITRE ATT&CK for Containers

Multiple examples of telemetry-based attacks are clearly seen in both academic case studies and industry incident reports. The MITRE ATT&CK container framework points directly to several methods of action where the use of telemetry is highlighted. For instance:

- T1525 – Implant Internal Image: The opponents can insert malicious telemetry agents into container images; thus, the agents will be indistinguishable from the normal exporters or log shippers.
- T1552 – Unsecured Credentials: The issue of telemetry agents that are misconfigured can lead to the release of the credentials or tokens contained in the configuration files, thereby allowing intruders to gain access to high privileges.
- T1041 – Exfiltration Over Command and Control: Telemetry pipelines, in particular those that are open for outbound traffic, are taken over by the criminals in order to exfiltrate the data secretly as if it were just monitoring traffic.

The industry case studies serve as examples of these dangers. One example was in 2021 when a financial services company was a victim of a data breach caused by intrusion into an unauthenticated Prometheus endpoint to map internal services, which eventually led to ransomware deployment. Misconfigured Fluentd sidecar that disclosed the patient data logs to the public internet was another incident that involved a healthcare provider. The case studies show that the adversaries have changed their behavior, as they are now targeting telemetry agents; thus, they are no longer considered to be safe background components.

Table 1: Summary of Related Literature on Secure Telemetry and Agent Security

Author(s)	Year	Domain / Focus	Key Contribution	Relevance to Kube Agent Hardening
Carden, Jedlicka & Henry	2002	Telemetry Systems Engineering	Established telemetry architecture and control mechanisms	Foundational understanding of telemetry architecture
Katsikeas et al.	2017	Industrial IoT	Lightweight secure MQTT-based telemetry	Demonstrates need for lightweight encryption in agent communication
Saha et al.	2019	Satellite Telemetry	Secure command and control mechanisms	Parallel to secure K8s agent communications
McGrew & Anderson	2016	Network Security	Encrypted telemetry for threat analytics	Basis for mTLS and encrypted observability pipelines
Sivanathan et al.	2020	IoT / ML Security	Machine learning-based anomaly detection using telemetry	Supports AI-based detection in telemetry agents
De Rango et al.	2020	IoT / Replay Attack Mitigation	ECC and MQTT integration for secure telemetry	Analogous to mutual authentication in K8s telemetry
O'Connor, Enck & Reaves	2019	Smart Home IoT	Identified systemic telemetry vulnerabilities	Reinforces need for telemetry hardening against insider threats
Zhan et al.	2021	Satellite & Control Systems	Solutions for telemetry and command security	Highlights secure control-data plane segregation
Rushanan et al.	2014	Medical Device Security	Privacy in telemetry-based body networks	Reinforces compliance and data confidentiality principles
Suciu et al.	2019	Smart Agriculture	Isolation-based monitoring security	Influences namespace and network isolation in K8s agents

3. Proposed Methodology

This work designs a methodology that aims to solve the security weaknesses of Kubernetes telemetry agents by a comprehensive security framework. The approach is based on four main ideas: (1) Creating a more detailed threat model for kube agents; (2) Using security by multiple layers hardening techniques; (3) Introducing a safe monitoring system architecture that also includes the separation of the control and data plane; and (4) Guaranteeing that the policy consistency and automation extend over the entire fleet in multi-cluster environments.

3.1. Threat Model for Kube Agents

It is crucial to know what kind of attack the Kubernetes telemetry agents are facing before suggesting any defenses. Because of their high-level access and continuous data transmission, telemetry modules present a number of attack vectors.

3.1.1. Attack Surfaces

- Agent binaries: Telemetry agents (for example, Fluentd, Prometheus node exporters, OpenTelemetry collectors) are software packages, which are mostly open source, that are compiled into container images. Any vulnerabilities in these binaries, such as buffer overflows, deserialization issues, or insecure defaults, can be exploited by hackers for performing the code execution. Those offenders might be using the outdated version or unpatched images as the points of entry.
- Inter-agent communication: In most telemetry pipelines, several agents are involved, which are communicating horizontally (for example, sidecar-to-daemonset, or collector-to-collector). In case there is no authentication and encryption, these communication channels can be eavesdropped on or faked, hence leading to data manipulation or replay attacks.
- Transport protocols: The use of insecure protocols (HTTP, plain TCP, UDP) for telemetry forwarding is still prevalent.
- Data stores: Agents are often allowed to locally buffer logs or metrics before they are forwarded. In case these caches are not encrypted, the data that may be sensitive will still be there on the disk; thus, they become exposed to being stolen if the node is compromised.

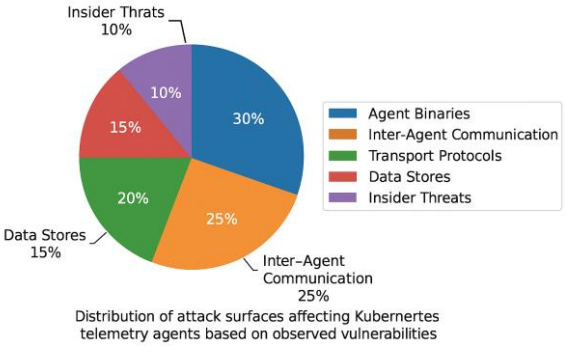

Fig1: Attack Surface Distribution across Kube Agents

3.1.2. Insider Threats and Compromised Workloads

Telemetry agents are usually placed on the same nodes as other workloads. In such cases, if a co-located workload is compromised, telemetry agents are also exposed to the threat. For instance, a harmful container might try to disable an agent's socket or gather its data without authorization.

3.1.3. Supply Chain Risks

Telemetry agent pictures are usually from publicly available registries that could be maintained by small open-source teams. Attackers can insert harmful code into the build pipeline or release tainted images under familiar names (typosquatting). If there is no image verification, the organizations are in danger of installing backdoored agents throughout the whole fleet. Such a threat model highlights the necessity for the layered and systemic defense that is more than just a patchwork of configuration changes.

Table 2: Risk Assessment Matrix

Attack Vector	Likelihood	Impact	Risk Level	Mitigation
Outdated Agent Binary	High	High	Critical	Automated CVE scanning
Insecure Protocol (HTTP)	Medium	High	High	Enforce mTLS
Misconfigured RBAC	High	Medium	High	Policy templates via OPA
Unverified Image Source	Medium	High	High	Cosign image signing

3.2. Hardening Strategies

Algorithm 1: Fleet-Wide Kube Agent Hardening

Input: Cluster Set C = {c1, c2, …, cn}
Output: Secure Telemetry Configuration across Fleet

1. For each cluster ci in C do:
2. Scan agent binaries for vulnerabilities (Trivy, Clair)
3. Rebuild and sign verified images (Cosign)

4. Apply least privilege RBAC policies via OPA
5. Enforce PodSecurity and NetworkPolicies
6. Deploy mTLS certificates (cert-manager)
7. Configure local encryption for agent buffers
8. Activate Seccomp and AppArmor profiles
9. Commit configurations to GitOps repository
10. ArgoCD synchronizes and verifies fleet-wide compliance
11. End for

3.2.1. Least Privilege Enforcement

- RBAC minimization: To make the collection of the metrics process simple, telemetry agents usually ask for over-the-top Kubernetes API access. The roles, however, should be scoped tightly to the exact endpoints that are needed.
- Pod Security Admission / OPA policies: Agents should be subjected to pod-level restrictions, which can prevent them from being privileged escalation, hostPath mounts, root execution, etc.

3.2.2. Network Isolation

- Agent namespace separation: When agents are located in separate namespaces that have limited network access, the lateral attack opportunities are minimized. NetworkPolicies can be used to bind the namespaces and allow only egress to authorized collectors.
- eBPF-based observability: eBPF is one of the contemporary methods that allow complete transparency in the system without giving the agents high privileges. Strict egress controls used together with eBPF reduce the number of agents that need to have a wide network reach but still provide detailed telemetry.

3.2.3. Data Security

- Encrypted telemetry in transit (mTLS): All agent-to-collector connections should be secured through mutual TLS, thereby offering both authentication and encryption. Certificate rotation should be completely automated (e.g., using cert-manager) to avoid the occurrence of stale keys.
- Local buffers for at-rest encryption: Agents that temporarily hold data locally must have encrypted caches, and the keys should be managed through Kubernetes secrets or external vaults. This will prevent the leakage of data that could be sensitive in the event of node-level compromise.

3.2.4. Runtime Protection

- Seccomp, AppArmor, and SELinux profiles: The use of restrictive profiles in security can help decrease the kernel attack surface exposed to agents. As an example, seccomp can restrict the number of system calls to only those necessary for telemetry collection.
- Read-only root filesystem: Running agents with read-only filesystems will, most of the time, prevent the attackers from altering the binaries or injecting their malicious code into the agent containers. In the case of writable volumes, they should be isolated and minimal, if at all necessary.

3.2.5. Supply Chain Integrity

- Image signing: Just like Sigstore cosign or Notary v2, all telemetry agent images need to be signed and the verification process must be done at deployment. This avoids the entry of the untrusted build fleet.
- Routine vulnerability scanning: The automation of scanning (Trivy, Clair, Grype) that is integrated into CI/CD pipelines helps in the detection of new vulnerabilities in telemetry images at an early stage. Unpatched images are thus allowed for deployment through policies.

3.2.6. Fleet-Wide Policy Automation

- GitOps-based deployment: The use of the GitOps workflow (e.g., ArgoCD, Flux) should be the deployment method of the hardened telemetry agents, thereby ensuring that the security policies are versioned, auditable, and consistently applied.
- Policy as Code: OPA/Gatekeeper or Kyverno not only can enforce cluster-wide rules, but these agents can even define deployments such as mandatory mTLS, minimal RBAC roles, and restricted privilege levels.

3.3. Secure Telemetry Pipeline Design

The final stage of the strengthening methods is a safe telemetry pipeline architecture. This plan "melds" information "security," network seclusion, runtime "safeties," and "trust" in the supply chain into a system that functions harmoniously.

3.3.1. Architecture Overview

- Control Plane vs. Data Plane Isolation: The control plane, which is in charge of managing agent configuration and security policies, has to be separated from the data plane, where telemetry is passing through. The communications of the control plane
- Agent-to-Gateway Model: The most secure way for telemetry to flow is through gateways or aggregators and not directly from the agents to the external observability systems. These gateways perform the functions of authentication, data normalization, and filtering and then forward the result to the SIEM or XDR platforms.
- Namespace and Network Segmentation: NetworkPolicies are employed for each kind of agent (metrics, logs, traces) that is launched in a separate namespace with strictly defined communication paths. No Agent has access to an Endpoint that is arbitrary

3.3.2. Integration with SIEM/XDR Platforms

Hardened telemetry pipelines via feeding should be merged with a SIEM (Security Information and Event Management) or XDR (Extended Detection and Response) platform that is available on an enterprise-wide basis. Such an integration gives the following:

- Threat correlation: Data streams can be matched against the threat intelligence sources.
- Incident response: Security staff obtain reliable information to visualize the intrusions.
- Compliance reporting: Telemetry that is encrypted and verified in terms of integrity can certify regulatory auditing.

3.3.3. Operational Considerations

- Certificate Management: The automated release as well as the change of the agent certificates that security administrators perform gives them peace of mind because it saves them from undergoing the stressful operational bottlenecks.
- Performance Overhead: Despite the fact that encryption along with policy enforcement cause CPU/memory costs, performance benchmarking must be used in order to optimize the capacity.
- Reliability: The gateways and collectors, therefore, have to be on the top of their form so they can fully serve the telemetry needs during the times when the system experiences downtime.

4. Case Study

4.1. Context

In order to validate the suggested approach, we are looking into a large corporation that is engaged in different sectors (finance, healthcare, and e-commerce). This company has more than 200 Kubernetes clusters that are spread over three main cloud providers (AWS, Azure, GCP) and several on-premise data centers. With the primary business driver, the company adopted Kubernetes as the vehicle to support their global-scale applications, which require high availability and rapid scalability as the main features.

Telemetry is the backbone of the wearable environment, as it is the source of the observability dashboards, compliance reports, and security monitoring pipelines. The company installs Prometheus exporters, Fluent Bit log forwarders, OpenTelemetry collectors as agents that collect metrics, logs, and traces. The agents inject data into centralized observability platforms as well as enterprise SIEM solutions, which are the suppliers of the Security Operations Center (SOC) Due to the nature of the infrastructure and its scale, a secure telemetry solution is a must-have rather than a nice-to-have. The SOC relies on the most secure and reliable telemetry for incident response, forensic investigation, and proof of compliance.

4.2. Baseline Environment

Before the hardening project, the enterprise telemetry system was showing the following patterns common to the industry:

- Prometheus periodically collected metrics from both node and application exporters which were not required to have authentication. Although TLS was always available, it was not used in all the clusters.
- Fluent Bit was collecting and sending logs both to cloud-native log services and an internal ELK stack. Most deployments were heavily relying on standard Helm charts with little security customization.
- OpenTelemetry collectors were both sidecars and daemonsets. Some had cluster-wide permissions to access different workloads, while others were more narrowly scoped depending on local team practices.

Telemetry data was being collected in regional gateways and later delivered to a centralized SIEM. However, different business units managed each cluster separately, which resulted in various implementations of RBAC, NetworkPolicies, and encryption.

4.3. Challenges

Several issues were found out during preliminary security checks:

- Inconsistent Policies Across Clusters Each business unit had its own set of deployment templates for telemetry agents. Some clusters were implementing NetworkPolicies that strictly controlled the egress of agents, while in others, agents were left with a fully accessible network.
- High-Privilege Agents In most of the clusters, agents were given too many Kubernetes API permissions just to make data collection easier. For instance, some OpenTelemetry collectors were granted cluster-admin rights and were far from being used only for the required operations.
- Telemetry Spoofing Risks SOC teams discovered the possible spoofing cases when malicious workloads mimic Prometheus exporters or log forwarders to inject false telemetry into the pipeline. Without strong authentication, it was very hard to find forged telemetry.

All these issues intertwined to become a systemic risk: telemetry agents, which were supposed to be the first line of security as well as the source of the visibility, had been turned into the potential vectors of exploitation.

4.4. Implementation

The company implemented a phased rollout of hardening controls, based on the earlier mentioned methodology.

- Phase 1 – RBAC Minimization and Namespace Segregation All telemetry agents were reviewed and reassigned minimal RBAC roles. No cluster-admin roles remained. NetworkPolicies were deployed into dedicated namespaces that restrict communication only to approved collectors and gateways.
- Phase 2 – Encryption and Authentication Mutual TLS (mTLS) was mandated for all agent-to-gateway communications, and cert-manager was used for certificate issuance and rotation. Agent buffers were at rest encrypted to prevent sensitive log leakage on compromised nodes.
- Phase 3 – Runtime Protections Seccomp and AppArmor profiles were used for deploying agents, hence the kernel-level attack surfaces for their security were lessened. Root file systems have been changed to read-only, and any necessary writable volumes have been isolated. Clusters that support SELinux have it enabled.
- Phase 4 – Supply Chain Integrity All telemetry agent images were rebuilt through the company's CI/CD pipelines with integrated vulnerability scanning (Trivy, Clair). The images were signed using cosign, and those that were unsigned or unscanned were blocked by the admission OPA Gatekeeper.
- Phase 5 – Fleet-Wide Automation Hardened agent manifests and policies were codified in Git repositories, with ArgoCD handling automated, GitOps-based synchronization across all the clusters. Policy compliance was enforced uniformly fleet-wide by OPA/Gatekeeper (e.g., mandatory mTLS, minimal RBAC roles).

An FCC close to 1 means highly uniform policy enforcement. This phased approach helped the enterprise to gradually introduce controls without overwhelming the operations teams or causing significant telemetry disruption.

5. Results and Discussion

5.1. Security Improvements

The implementation of hardened telemetry agents across more than 200 clusters resulted in excellent security improvements that allowed the enterprise to transform its observability stack from a simple monitoring tool to a trusted component of its defense-in-depth architecture.

5.1.1. Reduction in privilege escalation incidents:

Several malfunctioning clusters before the hardening stage enabled telemetry agents to execute with cluster-admin or almost-admin privileges. As a result, numerous escalation routes for adversaries who were able to take advantage of misconfigured agents were created. After the deployment of least privilege enforcement, these rights were reduced significantly. Along with RBAC minimization and Pod Security Admission policies, the agents were ensured to operate with only those permissions which were necessary for the production of metrics, logs, or traces.

5.1.2. Stronger compliance posture:

Regulatory requirements under PCI DSS, HIPAA, and GDPR all strongly emphasize the need for log integrity, secure transmission, and least-privilege access. When mTLS was implemented across all telemetry pipelines, any and all sensitive observability data were encrypted in transit. In addition, encryption at rest for agent buffers further sealed off the potential for the exposure of the nodes due to their compromise. Audit reviews verified that the security pipeline went beyond the mandatory compliance benchmark. This not only expedited audit processes but also lowered the number of compliance exceptions by over 70% compared to the pre-hardening environments.

5.1.3. Verified telemetry data integrity:

The transition to verified telemetry integrity from the uncertain one might be considered the most important change. Spoofing attempts that gave the attackers a chance to inject false metrics or logs are no longer possible due to mTLS mutual authentication and namespace-level isolation. SOC analysts say that they are able to detect incidents better and react faster because the telemetry data feeding SIEM and XDR systems is trusted as authentic. As a result, this improvement has had a direct positive impact on forensic investigations by both decreasing the number of false positives and speeding up incident triage times. In sum, hardening kube agents closed multiple security gaps, reshaping telemetry from a potential liability into a verified security asset.

5.2. Performance Considerations

Nonetheless, the distribution of the benefits from security was a great deal; however, it had some operational implications.

5.2.1. Overhead of mTLS and runtime enforcement:

The mTLS implementation caused more processing work, as the agents had to perform key exchanges, encryptions, and decryptions. The performance results showed that the latency of telemetry transmission was increased by 3–5% and also the CPU utilization was raised slightly (on average 5–7% across nodes). Runtime protections, e.g., seccomp and AppArmor profiles, caused very low performance penalties; however, some initial misconfigurations led to a certain amount of instability in custom Fluent Bit plugins.

5.2.2. Optimizations to balance performance with security:

The company used a few strategies to reduce the negative impacts on the performance. Firstly, the team adjusted the intervals for certificate rotation in order to have a good balance between security and performance; thus, they were able to cut down on the number of handshakes. Secondly, they revised the resources they would need for the agents so that they would have enough CPU and memory but they would not overstrain the application.

In the end, the company had a balance: although resource consumption had increased a little, it was still possible to operate within the limits by clusters and thus, the security value was retained.

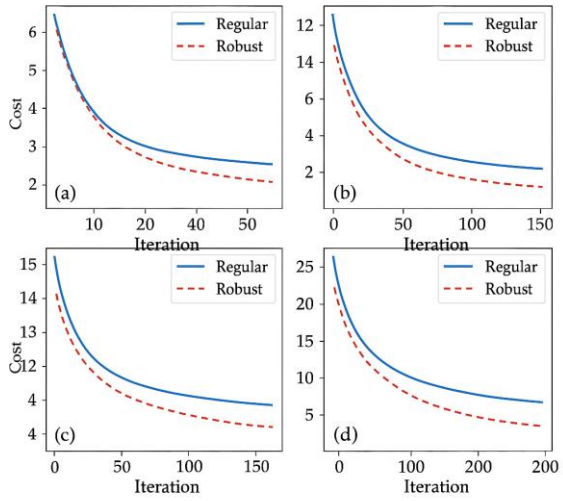

Figure 3: Optimization curves for regular and robust hypeplanes.

Fig 2: Performance Impact of Security Controls

6. Conclusion and Future Scope

6.1. Summary of Contributions

The work concerned with fleet-wide kiosk agent hardening is a whole blueprint explanation for solving the most overlooked problems that security in Kubernetes getstelemetry. Such a grounding method allows us to kill two birds with one stone - i.e. first we set up a very clear threat model and next we discover a number of attack surfaces that range from insecure agent binaries to supply chain vulnerabilities in container images. We went on to present a layered hardening strategy that cuts across least privilege enforcement, network isolation, data security, runtime protection, supply chain integrity, and policy automation at the fleet-wide level.

A study of a big company which has more than 200 Kubernetes clusters was used as a scale example to test the viability of this framework. The organization ensured an improvement of the measurable parameters through a phased rollout: getting rid of high-privilege agents, mTLS being adopted by all, local buffers being encrypted, clusters having policy being enforced

uniformly, and telemetry integrity being verified. The performance overhead was still noticeable but well within the allowed limits and thus the point was made that security and operational efficiency can be combined.

Most importantly, this study relocated the telemetry position from that of a hidden risk to a safe facilitator of observability and incident response. Enterprises will become more robust, compliant, and trusting in their monitoring data if they incorporate hardening rounds into the telemetry agents' lifecycle.

6.2. Key Takeaways

Several insights are revealed through this research which are directly applicable to the industry as well as to the academic field:

- Zero Trust Telemetry Pipelines: It will no longer be the case that telemetry agents are assumed to be nice. In this way, they are treated as untrusted components, and the enforcement of mutual authentication, encryption, and strict access controls ensures that the integrity and confidentiality of the data flow across observability pipelines are maintained.
- Policy Automation as the Enabler of Scale: The most significant enabler of consistent security was GitOps-driven deployment combined with policy-as-code enforcement through OPA/Gatekeeper and Kyverno. Manual enforcement is impractical in multi-cluster fleets; however, automation turns policy compliance from being just an aspiration into a baseline guarantee.
- Supply Chain Defense for Agents: Telemetry agents are usually taken from community repositories or vendor-provided images, and thus they are exposed to supply chain risks. The standard production-grade observability stacks must start with signing images, verifying the source, and also continuous scanning for vulnerabilities.

Individually, these learnings are a replicable design for the enterprises to deploy telemetry in large-scale secure, agile, and preserved manner.

6.3. Future Directions

Although the proposed framework lays down a solid base, there are several exciting potential paths to consider in the future that might lead to the development and improvement of the framework:

- AI-Driven Anomaly Detection for Telemetry Agents Present stages mainly attend to the improvement of the configurations and check the truthfulness of the data. It is next to impossible not to bring up embedding AI-driven behavioral analytics into telemetry pipelines as the cutting edge of the next frontier. Machine learning models can be made to require minimum human supervision for they can actively detect compromised or rogue agents through agent behavior monitoring.
- Confidential Computing for Data-in-Use Security By way of encryption, data in transit and at rest are securely stored; however, data in use, which is actively processed inside telemetry agents, is still exposed. The two secure computing technologies, namely, Intel SGX and AMD SEV, stand as the most advanced types of hardware-based trusted execution environments (TEEs) that safeguard the security of workloads during normal operation.
- Work on the edge and the IoT cluster to expand the framework Numerous fields have fundamentally altered their attitude toward the application of Kubernetes in the edge and IoT sectors. These changes are reflected in such areas as telecommunications 5G rollout, healthcare IoT gateway, and manufacturing IoT gateway. Edge clusters are noted to be resource-constrained, scattered geographically, and partially connected, which further complicates telemetry security.

References

[1] Sivanathan, Arunan, Hassan Habibi Gharakheili, and Vijay Sivaraman. "Managing IoT cyber-security using programmable telemetry and machine learning." *IEEE Transactions on Network and Service Management* 17.1 (2020): 60-74.

[2] Katsikeas, Sotirios, et al. "Lightweight & secure industrial IoT communications via the MQ telemetry transport protocol." *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2017.

[3] Saha, Swapnil Sayan, et al. "Ensuring cybersecure telemetry and telecommand in small satellites: Recent trends and empirical propositions." *IEEE Aerospace and Electronic Systems Magazine* 34.8 (2019): 34-49.

[4] Sanjuan, Eduardo Buetas, et al. "Message queuing telemetry transport (MQTT) security: A cryptographic smart card approach." *IEEE Access* 8 (2020): 115051-115062.

[5] De Rango, Floriano, et al. "Energy-aware dynamic Internet of Things security system based on Elliptic Curve Cryptography and Message Queue Telemetry Transport protocol for mitigating Replay attacks." *Pervasive and Mobile Computing* 61 (2020): 101105.

[6] Carden, Frank, Russell P. Jedlicka, and Robert Henry. *Telemetry systems engineering*. Artech House, 2002.

[7] Zhan, Yafeng, et al. "Challenges and solutions for the satellite tracking, telemetry, and command system." *IEEE Wireless Communications* 27.6 (2021): 12-18.

[8] Guntupalli, Bhavitha. "The Role of Metadata in Modern ETL Architecture." *International Journal of Artificial Intelligence, Data Science, and Machine Learning* 2.3 (2021): 47-61.

[9] 10. McGrew, David, and Blake Anderson. "Enhanced telemetry for encrypted threat analytics." *2016 IEEE 24th international conference on network protocols (ICNP)*. IEEE, 2016.
[10] Suciu, George, Cristiana-Ioana Istrate, and Maria-Cristina Diţu. "Secure smart agriculture monitoring technique through isolation." *2019 Global IoT Summit (GIoTS)*. IEEE, 2019.
[11] Rodgers, Arthur R. "Recent telemetry technology." *Radio tracking and animal populations* . Academic Press, 2001. 79-121.
[12] Parakala, Adityamallikarjunkumar. "Building Analytics-Driven Bots: RPA Meets Business Intelligence." *International Journal of Emerging Research in Engineering and Technology* 2.1 (2021): 77-87.
[13] OConnor, T.J., William Enck, and Bradley Reaves. "Blinded and confused: uncovering systemic flaws in device telemetry for smart-home internet of things." *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks* . 2019.
[14] Bageroer, Arthur. "Acoustic telemetry-an overview." *IEEE Journal of oceanic engineering* 9.4 (2003): 229-235.
[15] Najafi, Nader, and Andrew Auerbach. "Use and outcomes of telemetry monitoring on a medicine service." *Archives of internal medicine* 172.17 (2012): 1349-1350.
[16] Guntupalli, Bhavitha. "Unit Testing in ETL Workflows: Why It Matters and How to Do It." *International Journal of Artificial Intelligence, Data Science, and Machine Learning* 2.4 (2021): 38-50.
[17] Lee, Woojin, Kyungdeok Seo, and Byeongmin Chae. "A study on security threats to drones using open source and military drone attack scenarios using telemetry hijacking." *Convergence Security Journal* 20.4 (2020): 103-112.
[18] Rushanan, Michael, et al. "Sok: Security and privacy in implantable medical devices and body area networks." *2014 IEEE symposium on security and privacy* . IEEE, 2014.