How Runtime Sensors Work

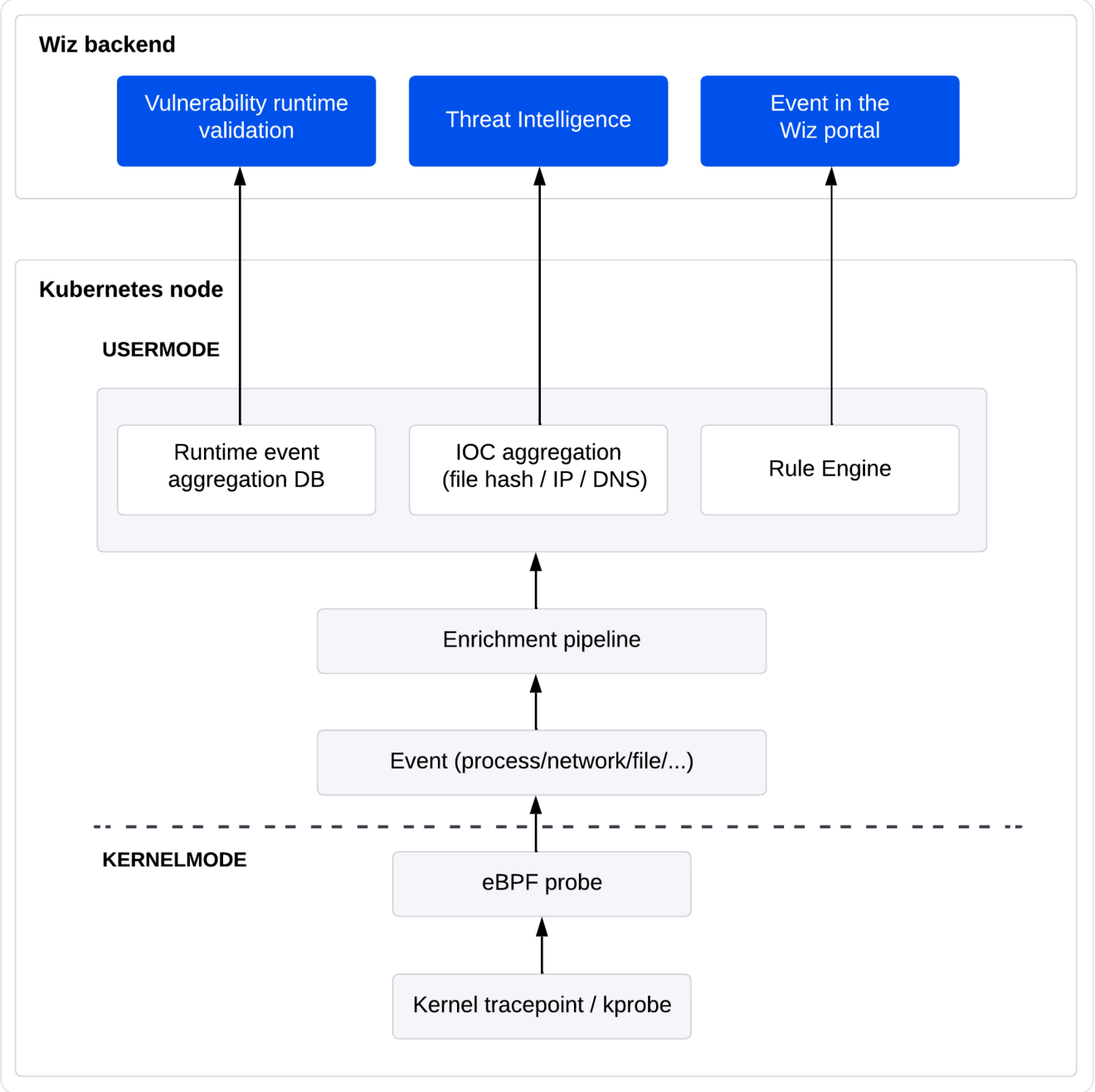
This feature requires a Wiz/Gov Advanced license. Learn more.

The Wiz Runtime Sensor is a Linux-based executable that provides visibility into runtime workloads in cloud and on-premises environments. Once deployed, the Runtime Sensor monitors all processes and containers on all nodes. This allows it to validate vulnerabilities in runtime and detect threats in real time. The Runtime Sensor reports its detections back to Wiz, where they are represented as cloud events, enriched with details on all relevant entities.

The Runtime Sensor uses eBPF probes to monitor the system state. It is lightweight in memory and compute footprint, and designed to be maximally compatible and non-disruptive to modern production workloads.

The Runtime Sensor can be installed on both cloud and on-prem Kubernetes clusters. It is deployed as a DaemonSet. Once deployed, it monitors all containers and processes on the node and performs runtime analysis. The Runtime Sensor communicates with Wiz to report status, pull updates, and send findings.

Architecture



The Runtime Sensor monitoring workflow is described in the diagram above. The important steps in this workflow are:

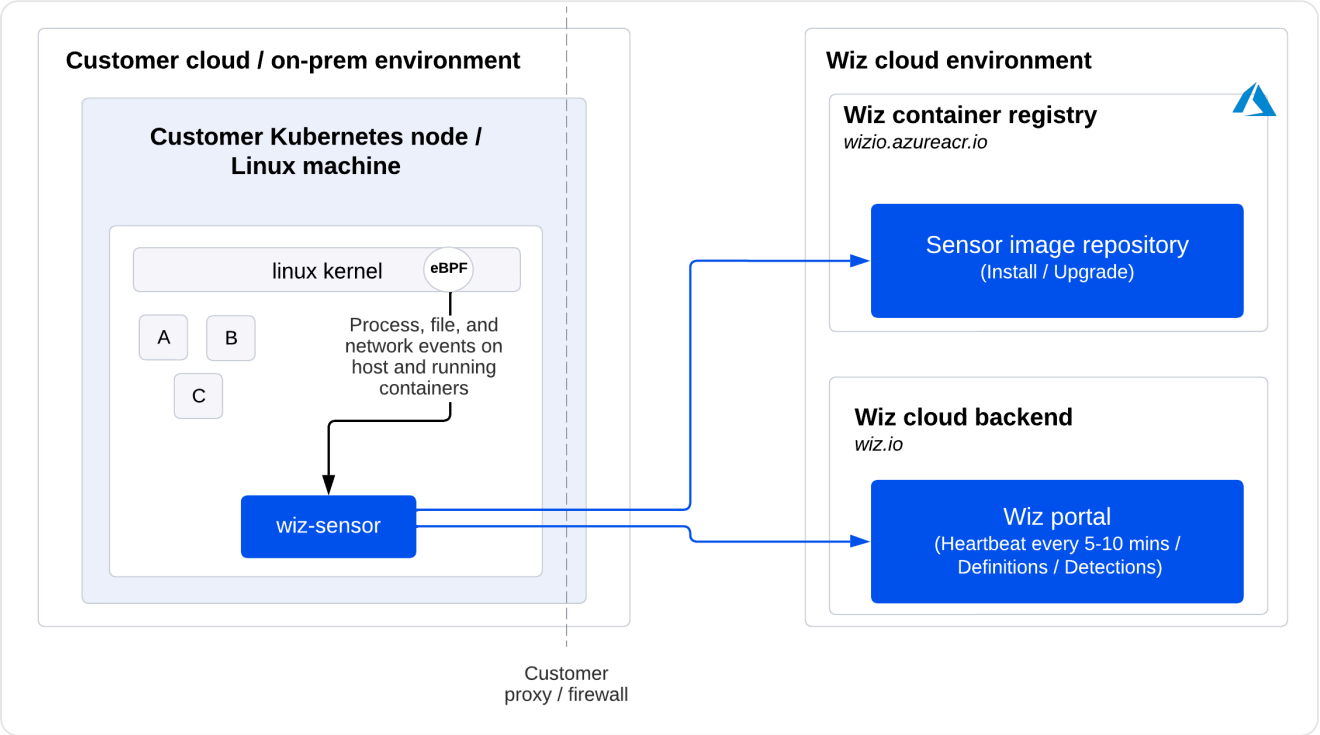
1. Every eBPF probe sends a message to the Runtime Sensor executable. A message is turned into an event.
2. Each event is enriched with context which is unavailable in kernel mode due to timing, execution, or memory constraints.
3. The enriched event is ingested by one or more of the relevant consumers:
    i. Runtime event aggregation database–Events relevant to the aggregated state of the node or container (module-load, process-start, etc.) are added to a local database. This database is periodically sent to the Wiz portal.
    ii. IOC aggregation–Events containing potential indicators of compromise (IOCs) such as process hashes, IP addresses, or DNS domains are sent to the Wiz Threat Intelligence service.

iii. Rule Engine–All Runtime Sensor events are sent to the Rule Engine for potentially malicious behavior matching.

Resource consumption

The Wiz Runtime Sensor is lightweight in memory and compute footprint, and designed to be maximally compatible and non-disruptive to modern production workloads. Learn more.

Communication with Wiz



The Runtime Sensor communicates with wiz.io and wizio.azurecr.io over the network, either directly or by using a proxy, through gRPC, TLS encrypted over HTTP/2 for synchronous communication, and REST HTTP over TLS for asynchronous communication. As the Runtime Sensor communicates with the Wiz backend directly, it is agnostic to Wiz SaaS or Outpost Deployments.

The main Runtime Sensor communication flows depicted in the above diagram are:

- Pull image–The Runtime Sensor is installed by the native Kubernetes docker-client, which pulls the Sensor image from an image repository and uses the image in a new Kubernetes pod on each node. The same method is applied for both initial deployment and ongoing upgrades of the Sensor.
- Send periodic heartbeats–The Runtime Sensor sends periodic heartbeats to the Wiz portal in order to report health status and also configuration details like definitions-version, proxy-settings, etc.

- Pull definition updates–The Runtime Sensor pulls new definitions or settings from the Wiz portal when those are available.
- Send detections–When the Runtime Sensor detects malicious behavior on the node, whether via IOC matching or the Rule Engine, it generates a detection and sends it to the Wiz portal (where it is displayed as a Cloud Event).

> Learn how we keep the Runtime Sensor communication safe.

Runtime analysis

Vulnerability runtime validation

Detect vulnerable executables and libraries that are loaded into memory. This helps security teams focus on vulnerabilities that are used in runtime (as opposed to vulnerable binaries that are never used). Read more here.

Real-time threat detection

Detect in real-time the execution of malware or malicious behavior (i.e. overwriting the passwords file, reverse shell execution, etc.). Read more here.

Network outage resilience

In case of network outages in which the Sensor cannot communicate with the Wiz portal:

1. During the first minute, the Sensor retries portal communication several times.
2. If unsuccessful, the Sensor has following auxiliary mechanisms in place:
   - Rule-Engine detections–Every hour, the Sensor retries to communicate with the Wiz Portal. If no communication is established within 24 hours, detections are marked as stale and deleted from the local database.
   - IOC-based detections–After 24 hours of lack of communication, all IOCs (file hashes, IP addresses, and domains) collected by the Sensor are deleted from the local database, as they might have become invalid in the meantime. If one of these IOCs is encountered again in the future, the Sensor will consult the Wiz Reputation Service as usual.
   - Runtime Execution Data (RED)–RED is sent to the portal and also stored on the local disk for backup. If the Sensor cannot communicate with the Wiz portal, the Wiz agentless scanner will pick up the RED collected within the last 48 hours, ensuring no data is lost.

Data collection and storage

The Runtime Sensor collects the following data from the clusters it is running in:

- Kubernetes resource metadata–Various properties about pods, deployments, DaemonSets, and other Kubernetes resources, including the resource name, status, attached service account, security context, labels, and included containers.
- Container metadata–Various properties of the running containers on the host, including the container id, start-time, name, and container image.
- Process activity–Information about process executions, including user-id, username, PID, file name, and command line. The hash of the file associated with the process is calculated and collected.
- File activity–File operations, such as opening, creating, or modifying a file. The file metadata is also collected, including the file name and directory.
- Network activity–Information about network connections made by monitored processes, including the source and destination IP addresses and ports.
- System call activity–Information about various operating system calls (currently, `ptrace` and `unshare`).
- Host information–Various properties about the VM the Runtime Sensor is running on, such as the operating system version, the kernel version, the number of CPU cores, and the amount of RAM memory.

Currently, the Runtime Sensor does not store raw events. Instead, during runtime analysis, the data is aggregated and stored in Wiz in two main ways:

- Runtime event aggregation database–Events relevant to the aggregated state of the host or container (module-load, process-start, etc.) are added to a local database on the Runtime Sensor host. This database is periodically sent to Wiz and used in vulnerability runtime validation.
- Threat detection database–When the Runtime Sensor detects malicious activity, it generates a detection that contains information about the suspicious event (i.e. process-start, file, network, or system call) and the associated process tree. These detections are stored in Wiz as cloud events for a period of 30 days.

> Before sending data to the Wiz backend, the Runtime Sensor redacts the following PII:
>
> - Secrets/passwords/tokens
> - Certificate data
> - Email addresses
> - JWT tokens
> - Auth URLs

Designed for scale

We've taken the following measures and design considerations to design the Runtime Sensor to work also on scale:

- The Sensor limits its Kubernetes API-server usage by:
  - Monitoring only the pods within its own node.
  - Featuring a random backoff mechanism for Sensor initialization to prevent API-server congestion in large clusters when multiple Sensors are initialized simultaneously.
  - Employing heavy caching when dealing with high-level controllers.
- The Sensor is lightweight in memory and compute footprint by design:
  - Ensuring minimal resource consumption for seamless integration with existing workloads.
  - Including a failsafe mechanism that, in the event of failure or lack of resources, results in a gradual loss of system visibility. Sensor resource shortage will never impact your environment. Any visibility loss is reported as a System Health Issue in Wiz for immediate monitoring and resolution.
  - Leveraging eBPF, a flexible instrumentation mechanism with strict controls on probe instruction counts. The Sensor only sets probes at locations that prevent the creation of system bottlenecks, like the exec() system call.
  - Featuring an adaptive alert aggregation system designed to suppress repetitive alerts at the Sensor level, helping maintain focus on significant events and reduce noise.
- The Sensor's egress restriction is achieved by setting limits on non-detection-related logs and telemetry, ensuring optimal performance by focusing resources on detection tasks.

Runtime Sensor security principles

The Wiz Runtime Sensor was designed from the ground up with security in mind. Multiple reinforcing measures help make it safe to install and use in your environment.

Learn about:

- Sensor security for Kubernetes
- Sensor security for Linux hosts

Updated 28 days ago

← Runtime Sensors

How Real-time Threat Detection Works →

English