Checklist: Container Security from Code to Runtime

Modern software development has shifted towards microservices built on containers, Kubernetes, and cloud, and at the same time, adoption of DevOps culture is driving continuous software deployment to rapidly meet business needs. Security in these fast-moving cloud-native environments requires a radically different approach.

Software vulnerabilities, misconfigurations, and suspicious activity at runtime are all areas of concern for developers, operations, and security teams. As containers and DevOps practices blur the boundary between code development and the production environment, security solutions that satisfy the needs of development teams or operations teams in isolation can leave gaps and result in inefficient security practices.

Adopting a cohesive and automated approach to security from development through production helps teams stay vigilant against cyberattacks, reduce noise, and tackle the unique risks of containers, Kubernetes, and cloud. The right practices from source to run are critical for securing your cloud-native environment, but will also enable greater efficiency to help you ship applications faster.

The following checklist outlines key security strategies and best practices to follow from source to run. These key aspects of container security center around three major themes:

Build Secure from the Start

Introducing security practices as early as possible in the development phase of your software development lifecycle (SDLC) helps you guard against issues that can expose risk, delay, and cost in later stages. This “Shift Left” approach encourages development teams to implement the required practices and tools to ensure they build secure applications from the start.

Protect against runtime threats

While code, container, and IaC security best practices provide protection from known issues and misconfigurations, these practices alone are not enough. A host of security threats, by their very nature, only manifest during runtime. Detecting and responding to malicious activity such as privilege escalation attempts in containers requires new vantage points and cloud-native controls.

Prioritize security alerts that matter

Containers are often bloated with contents and packages, overwhelming developers with vulnerabilities. Attempting to wade through an unmanageable number of issues takes precious time away from coding and leaves organizations open to risk. Techniques such as using runtime intelligence will help developers prioritize vulnerabilities for packages that are actually used when a container runs, reducing the burden by as much as 95%.

1

CONTAINER SECURITY FROM CODE TO RUNTIME

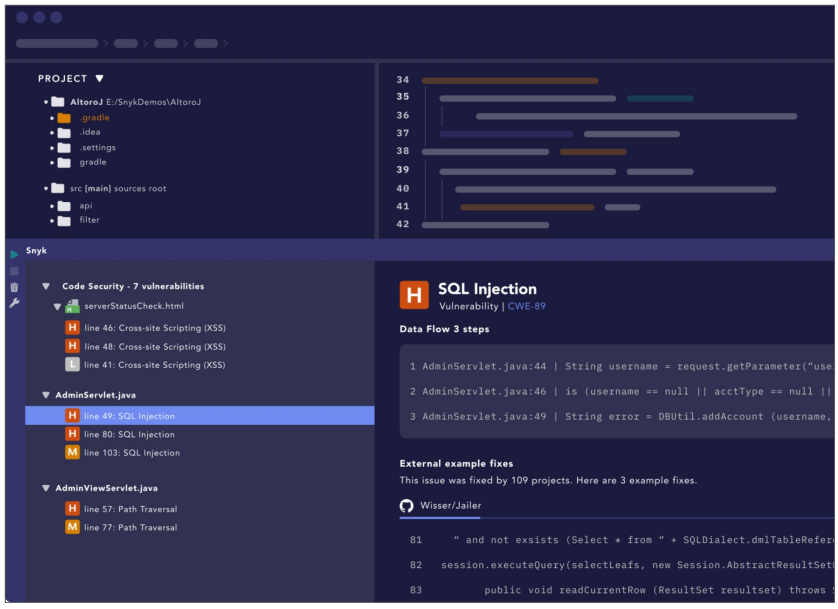
Code Security

Delivering your applications faster is likely one of the key reasons you've moved to containers in the first place. Even with containers, application security begins with your code. Source code is the aspect of containerized applications that is most directly controlled by developers. Tracking down all your code dependencies and figuring out how to fix security issues isn't trivial.

Using purpose-built code security tools to do software composition analysis (SCA) and static application security testing (SAST) to analyze your code and its dependencies will help you spot issues early in development. In addition, integrating these tools with your source code opens up the possibility to automate this process and catch these issues directly in your git commits and repositories.

Key aspects to look for when evaluating code security tools include:

- Real-time scan results to help you see results as you code.
- Integration with developer tools and workflows through plugins.
- Native Git scanning to test projects directly from the repositories.
- Daily monitoring for new vulnerabilities.
- Clear context and details for findings to help developers learn and prevent repeat issues.



sysdig snyk

2

CONTAINER SECURITY FROM CODE TO RUNTIME

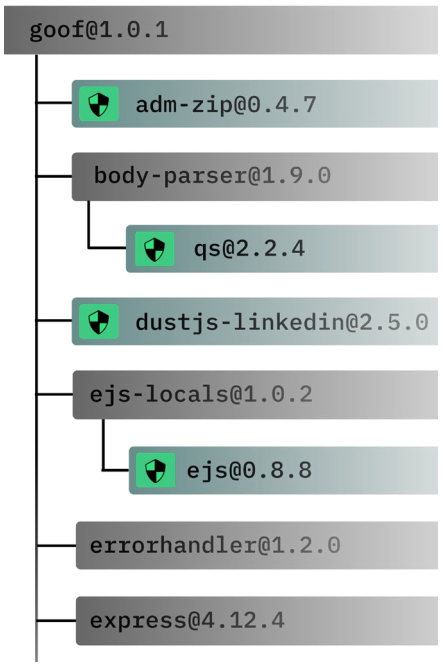
Open Source Security

In modern applications, it's not unusual for 3rd party open source dependencies to make up the majority of the lines of code in an application. Open source helps developers build faster. But how can you make sure your open-source dependencies are secure?

Development teams need to be able to find, prioritize, and fix security vulnerabilities and license issues in the open-source components they are using in their applications. Software composition analysis (SCA) tools for open source will help you scan for open source dependencies and flag any packages that are vulnerable. Ideally, the tool you use to identify open source issues will also integrate into the exact same processes that your developers use, such as your Integrated Development Environment (IDE) or a Git-based workflow.

In addition to flagging issues, leading solutions, such as Snyk, will also simplify the search for the most secure open-source packages available. A database of the more-than-a-million open-source packages helps you quickly identify the most appropriate fix. To evaluate the available open-source packages, a number of criteria are used including:

- **Popularity** — Understand the prevalence of an open-source package using metrics such as downloads and source code repository stars to measure popularity.
- **Maintenance** — Get insights about open-source dependency health and assess the sustainability of the project.
- **Community** — Is the community thriving for an open-source package you use in your project or has it gone stale? Gauge the status with project metrics.
- **Vulnerabilities and license security** — Assess the security posture of an open-source project and its past versions.



sysdig snyk

3

CONTAINER SECURITY FROM CODE TO RUNTIME

Image Security

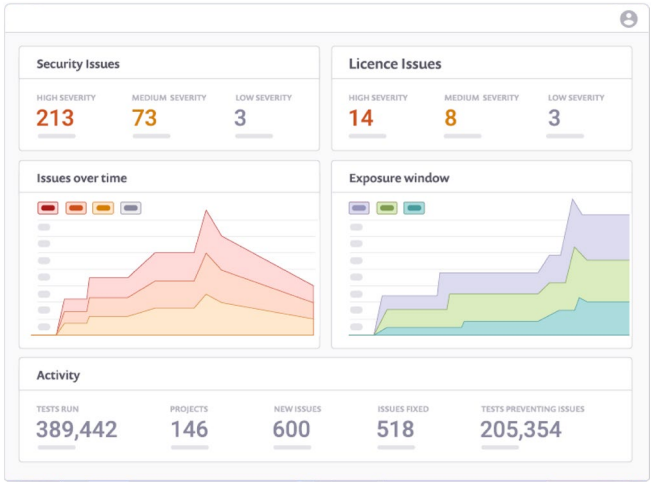
Containers provide a standard packaging format for applications, but container images can be opaque, leading to problems identifying the software and vulnerabilities they contain. There are a number of practices you can put in place to ensure the security of your container images from selecting the right base image to automating image scanning policies to help you efficiently identify and fix known vulnerabilities.

One of the most important considerations for container security is your base image. It's easy to go to a public registry like Docker Hub and find an image that matches your use case, but you need to pay attention to the provenance of the images you choose. Just like you wouldn't download and install software from an untrusted website, you likely would not want to use images from users you don't know and trust.

Many trustworthy vendors provide container images you can easily use, such as those from Verified Publishers on Docker Hub. While this provides some level of quality assurance, to further reduce the number of packages and potential vulnerabilities you should go a step further and choose minimal base images matched to your needs. Since you inherit whatever comes in the base image as you build up your own image on top of it, a slim image can reduce your security burden.

New vulnerabilities are disclosed continuously. To meet the challenge of this reality, choosing a container security solution that will alert you to new vulnerabilities in previously scanned images, including those running in production, is key to effective image security.

- Eliminate vulnerabilities by upgrading to a more secure base image.
- Use risk signals like exploit maturity and insecure workload configurations to cut through container vulnerability noise.
- Add automated scanning to your CI/CD pipeline.
- Monitor your running environment for newly disclosed vulnerabilities.
- Check for unsafe settings and image misconfigurations that could raise the risk of exploits and attacks.
- Automated base image fixes via native pull requests.



sysdig snyk

4

CONTAINER SECURITY FROM CODE TO RUNTIME

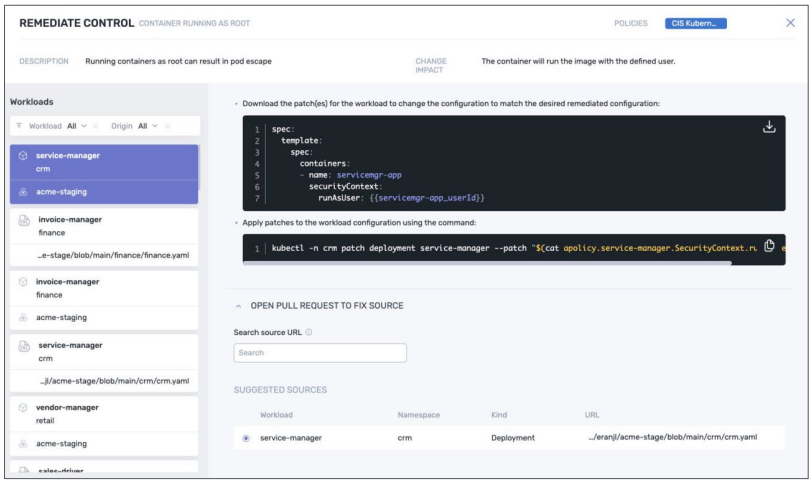
Infrastructure as Code Security

Modern cloud-native environments are defined, changed, and managed using infrastructure as code (IaC). In Kubernetes environments, for instance, teams can use declarative IaC configuration files, such as YAML, Terraform, and Helm to deploy consistent, and repeatable configurations. Similarly, IaC solutions like AWS CloudFormation, Azure Resource Manager, and Google Cloud Deployment Manager, model, provision, and manage cloud services and third-party resources for public cloud infrastructure.

IaC security focuses on detecting and fixing IaC misconfigurations such as overly permissive workload configurations across IaC templates using policy-as-code. It helps you strengthen both security, and compliance by auto-detecting and auto-remediating issues, including configuration drift. Drift occur when your intended environment configuration becomes different over time, usually due to manual changes and updates.

State-of-the-art IaC security tools will help you apply rules based on industry benchmarks and best practices and add custom rules.

- Test and monitor locally, in code repos, and as you plan and test your configurations in CI/CD pipelines.
- Implement fixes with a simple pull request and eliminate the guesswork of creating secure configuration in minutes.
- Configure and apply policies for compliance and governance across environments with a policy engine such as Open Policy Agent (OPA).



sysdig snyk

5

CONTAINER SECURITY FROM CODE TO RUNTIME

Runtime Security

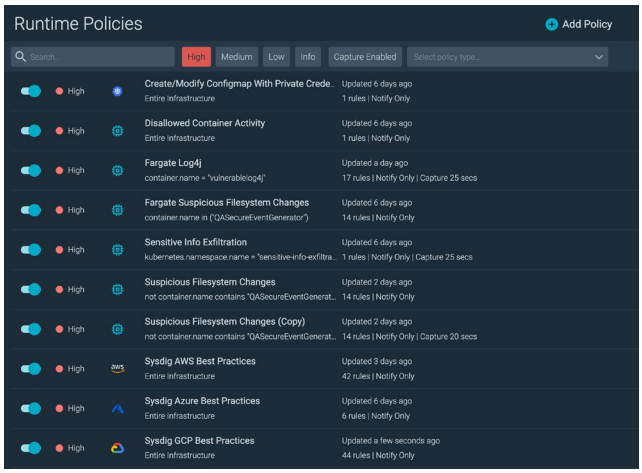
Runtime security solutions help you detect and respond to threats to your running containers. Operations and security teams need to know if there is any abnormal or unwanted behavior happening in or around container workloads. Monitoring container activity can be tricky since by their nature they are opaque, making it difficult to see inside. Without the proper tools, containers are black boxes.

Adding to challenge is the often short lifespan of containers. The recent Sysdig Cloud-Native Security and Usage Report found that 44% of containers live less than five minutes. To spot anomalies even during the short run of many container tasks requires real-time visibility. What you need is essentially the equivalent of a security camera that detects activity and captures the incident while alerting you to the event.

Attacks may exploit vulnerabilities that are not yet identified and access your environment in ways you haven't anticipated. Therefore, having an adaptable rules engine to help you cast a wide net and define policies to monitor activity across hosts, containers, and cloud is critical to staying on top of threats.

Top aspects to look for:

- **Light-weight instrumentation:** Gaining visibility inside containers should not require modification of your container images. A solution like open-source Falco, for instance, monitors container runtime activity from the Linux kernel by parsing system calls.
- **Event alerts with context:** Understanding "what" is happening, but also "where" helps you quickly address threats across complex cloud-native environments. Enriching event alerts with context such as cloud account, region, cluster, namespace, image, etc., helps you quickly pinpoint where to take action.
- **Pre-built and customizable policies:** Having out-of-the-box runtime security policies based on best practices will help you save time and get started quickly. Having the flexibility to tailor policies to your specific environment is also key to help you address risk for any use case.
- **Response actions:** Automate security event response aids in addressing risk without manual intervention. The ability to automatically take actions such as killing an impacted container to stop threats can help limit the impact of breaches and lateral movement attempts.



sysdig snyk

6

CONTAINER SECURITY FROM CODE TO RUNTIME

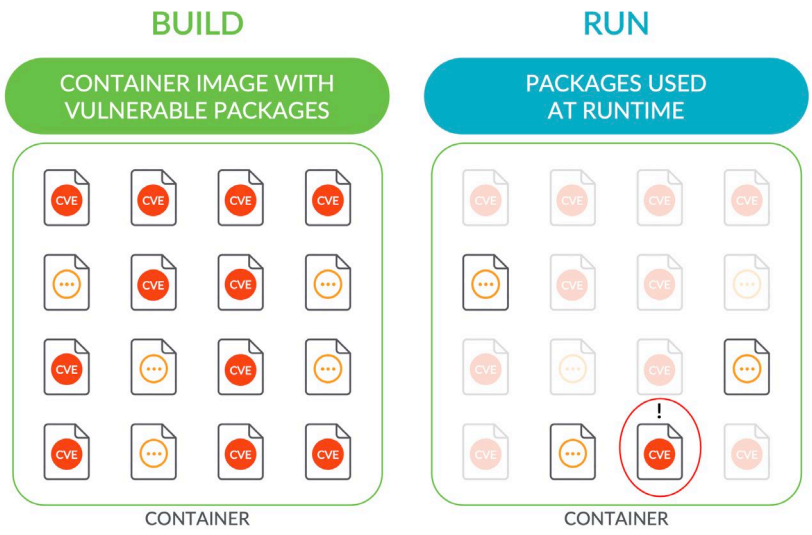
Vulnerability Prioritization Using Runtime Signals

As outlined above, developers are increasingly managing application security, from code and dependencies to the containers that ship and run. At the same time, security and operations teams working with live environments have to deal with the fallout from vulnerabilities present in production.

Developers find themselves overwhelmed with hundreds of vulnerabilities and don't know where to focus remediation efforts. Attempting to wade through an unmanageable number of issues is noise that takes precious time away from coding and leaves organizations open to risk.

For container images, there are often alternate images available that are more secure, updated, and slimmer. Choosing a "better" image alone can cut out 70% or more of initial vulnerabilities that would otherwise weigh down developers. But that still leaves a large number of the vulnerabilities — often numbers in the hundreds that can be a daunting task for developers.

One of the most recent advancements in prioritization techniques is to use runtime intelligence to identify the software packages actually executed in the running container to direct developers on what to fix first to address real risk. By bringing this information into the development pipeline, development teams can instantly eliminate up to 95 percent of the vulnerabilities that would otherwise demand their attention.



sysdig snyk

7

CONTAINER SECURITY FROM CODE TO RUNTIME

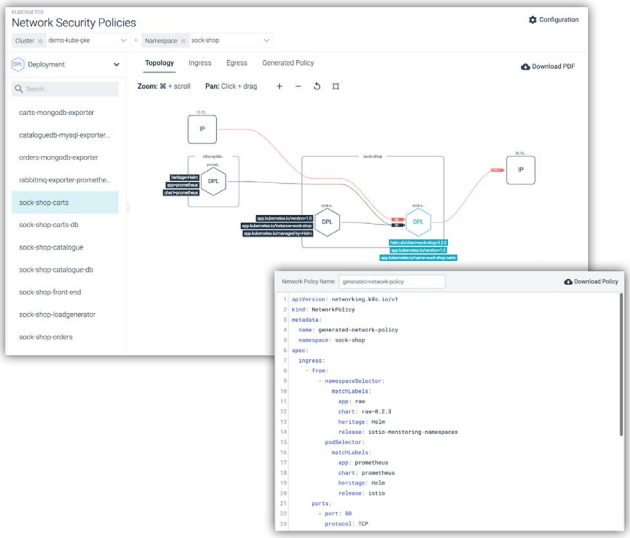
Network Security

Increasingly, enterprises are moving to a Zero Trust approach to network security, requiring all entities and users, whether in or outside the organization's network, to be authenticated, authorized, and validated for security configuration before being allowed access to applications and data.

In Kubernetes, pods are short-lived, they jump between hosts, have ephemeral IP addresses, and scale up and down. Trying to configure classic firewall rules is laborious, error-prone and an uphill battle since the environment is dynamic and constantly changing. For network security to be effective, you have to look beyond the physical communication layer.

The best strategy for network security with Kubernetes and containers is to use native controls, such as Kubernetes NetworkPolicy. Kubernetes network policies enable you to address the issue at the correct level of abstraction network segmentation. Yet, DevOps teams are often blind to how containerized apps are communicating. This makes it difficult to create effective policies. A lot of time can be wasted going back-and-forth between developers and ops to define the proper rules.

To properly implement container network security, you first need to have deep visibility into how containers and microservices are communicating with each other and what is required to function properly. The Kubernetes API provides metadata about namespaces, services, deployments, etc. and can help you begin to intelligently determine the proper rules for network communications at the Kubernetes level.



Creating these Kubernetes network security policies can still be a tricky process, requiring a relatively long YAML even for a simple app. To simplify this process, solutions like Sysdig Secure provide tools to simplify Kubernetes network policy creation. This includes:

- **Network topology maps:** Visualize all communication into and out of a particular pod, service, and application based on Kubernetes metadata.
- **Baseline network policies:** Auto-generated policies based on observed traffic enriched with application and Kubernetes metadata that you can directly refine and modify to match your desired declarative state.
- **Automated network policy generation:** Generate declarative network policy YAML files for network segmentation based on the topology baseline and any customized allowed ingress and egress adjustments.

sysdig snyk

8

CONTAINER SECURITY FROM CODE TO RUNTIME

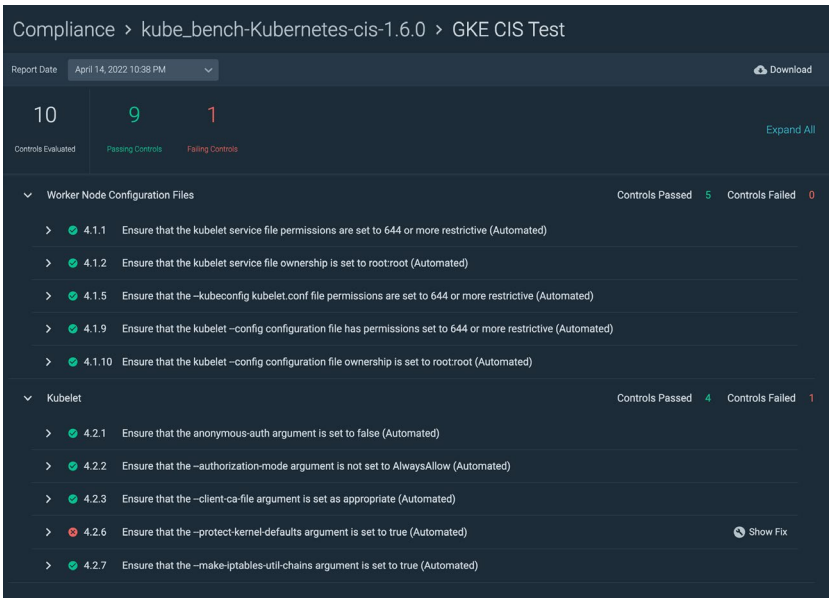
Kubernetes and Cloud Platform Security

Wrongly configured hosts, container runtimes, clusters, or cloud resources can leave a door open to an attack, or create an easy way to escalate privileges and perform lateral movement.

Benchmarks, best practices, and hardening guides provide you with information about how to spot misconfigurations, why they are a security problem, and how to remediate them. One source of this type of information is the Center for Internet Security (CIS), a non-profit organization that publishes free benchmarks for many different environments, including Kubernetes, and cloud. These benchmarks are based on research as well as contributions from the community. The information provided by CIS has become the de facto standard for security benchmarking.

The best way to make sure you can check settings for Kubernetes and cloud platform security is to automate it as much as possible. Several tools exist for this, mainly based on static configuration analysis, allowing you to check configuration parameters at different levels and providing guidance for fixing issues.

Mature solutions provide features that help you schedule, execute, and analyze a wide range of infrastructure including Linux hosts, Docker, Kubernetes, EKS, GKE, Openshift clusters, etc., as well as provide a lens into meeting compliance standards, like PCI DSS, SOC 2, NIST 800-53, NIST 800-190, HIPAA, ISO 27001, GDPR and others, all in a single centralized dashboard.



sysdig snyk

9

CONTAINER SECURITY FROM CODE TO RUNTIME

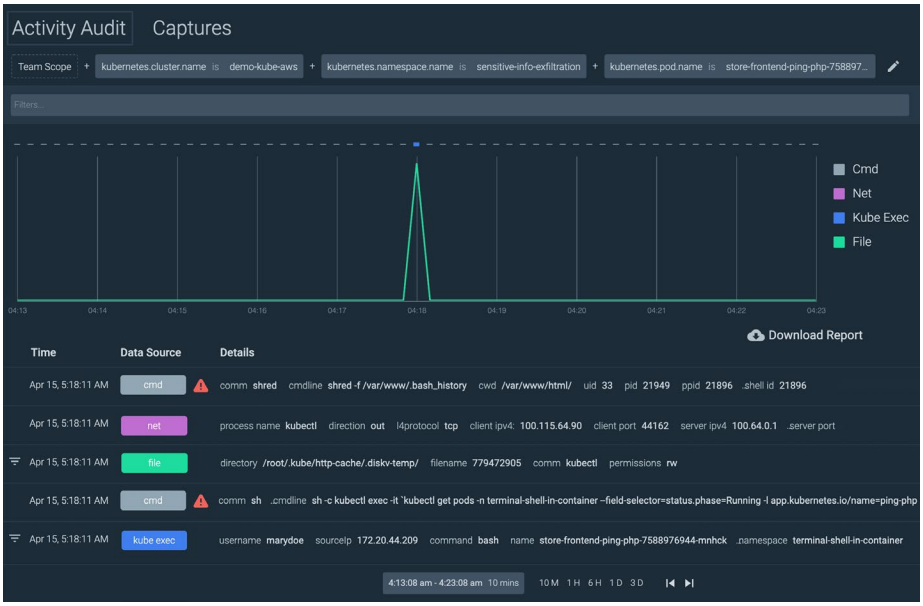
Incident Response and Forensics

When a security incident does occur, you'll need information surrounding the issue to investigate, respond, and prevent future occurrences. Given the dynamics of containers, they may prove elusive for traditional incident response approaches. For instance, by the time you're able to start your investigation, the containers may already be gone.

Will you have digital evidence of a security breach and be able to carefully analyze and respond to an attacks in a methodical and timely manner? Once you detect something suspicious, you may want to go in-depth and evaluate the risk of the event. The ability to record activity surrounding an event will ensure you can explore an incident even after the container or host is no longer alive. Sysdig Secure for instance records activity across containers, Kubernetes, and cloud, and captures syscall activity from which you investigate a wide-range of activity to get a complete picture, including:

- User activity.
- Suspicious commands.
- Altered files.
- Unexpected network traffic.

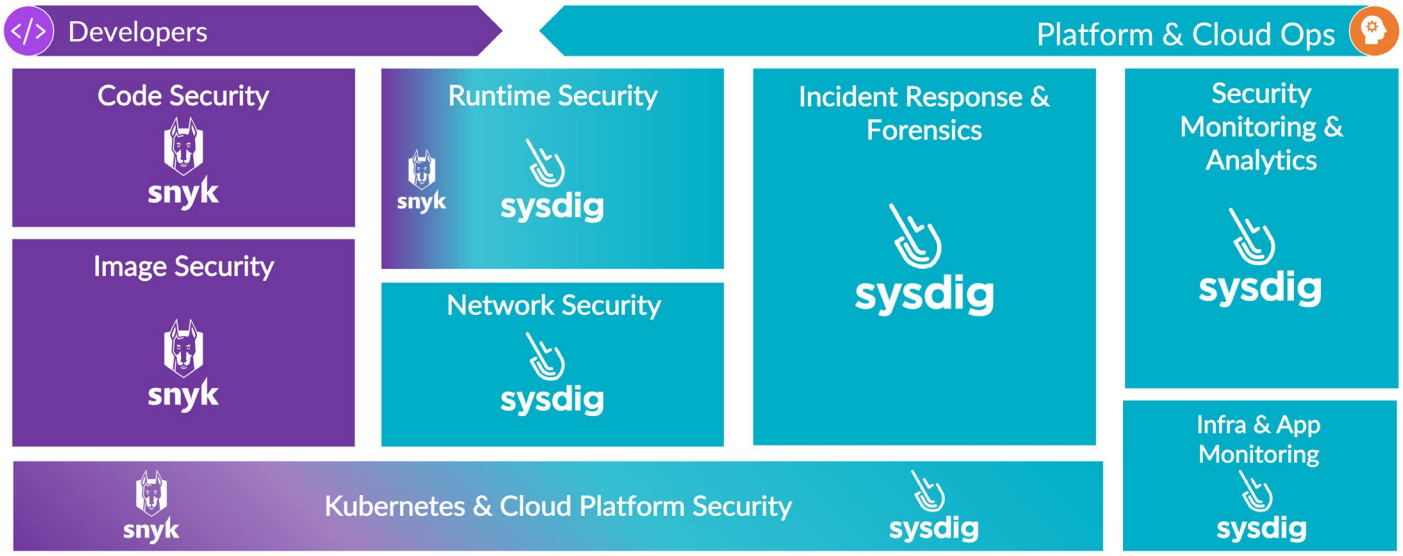
Having the right information at your fingertips will help you respond quickly so you can minimize the impact of an incident, take action to reduce the attack surface, and prevent future episodes.



sysdig snyk

Container Security from Code to Runtime with Snyk and Sysdig

Sysdig and Snyk bring together the industry-leading container runtime and developer security tools, for the first integration that bridges developer, DevOps, and SecOps silos. Sysdig's unique container visibility and threat protection and Snyk's developer-first tooling pair accurate runtime threat protection with early detection and vulnerability management. By bringing this information into the development pipeline, Sysdig and Snyk are in a unique position to help development teams instantly eliminate up to 95 percent of the vulnerabilities that would otherwise demand their attention.



Learn More