Brought to you by:

sysdig

Securing Containers & Cloud

for dummies®

A Wiley Brand

Protect cloud, Kubernetes, and containers

Manage configuration and permission risk

Prioritize vulnerabilities and detect threats

Daniella Pontes

Pawan Shankar

Sysdig Special Edition

About Sysdig

Sysdig is driving the standard for cloud and container security. The company pioneered cloud-native runtime threat detection and response by creating Falco and Sysdig as open source standards and key building blocks of the Sysdig platform. With the platform, teams can find and prioritize software vulnerabilities, detect and respond to threats, and manage cloud configurations, permissions, and compliance. From containers and Kubernetes to cloud services, teams get a single view of risk from source to run, with no blind spots, no guesswork, no black boxes. The largest and most innovative companies around the world rely on Sysdig.

Securing Containers & Cloud

Sysdig Special Edition

by Daniella Pontes and Pawan Shankar

for dummies®
A Wiley Brand

Securing Containers & Cloud For Dummies®, Sysdig Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-119-84345-0 (pbk); ISBN: 978-1-119-84346-7 (ebk). Some blank pages in the print version may not be included in the ePDF version.

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Manager and Development Editor: Carrie Burchfield-Leighton

Acquisitions Editor: Ashley Coffey

Sr. Managing Editor: Rev Mengle

Business Development Representative: Matt Cox

Authors' Acknowledgments

The idea to create this book was born from our commitment to knowledge sharing, a core value of our DNA. But it became possible because of how we embrace our goals as a team. We'd also like to thank Aaron Newcomb for his special contribution on the topic of monitoring.

Table of Contents

Foreword

Security is a top concern when running software applications and cloud infrastructure. Whether you're in the early stages or running all your workloads in production, you probably already noticed that cloud-native security is different from IT-managed data center security. The dramatic increase in complexity and the evolving threat landscape make cloud and container security even more critical and harder!

To develop and operate securely in the cloud requires addressing a core issue: blind spots. IT teams used to have direct access to and full control of any production resource, but now they have to rely on the limited visibility provided by the cloud provider's application programming interfaces (APIs). They are also not the only ones to provision and make changes to production as in the old days. The unprecedented distributed ownership, flexibility, and automation enabled in the cloud that make developers so productive in shipping apps fast are also a constant source of risk in the cloud. Misconfigurations of security controls, roles with excessive permissions, vulnerabilities in container and host images, and internal threats and evolved attacks that evade legacy security — all at the scale and speed of cloud — become critical blind spots used as attack vectors. Security in the cloud is a multidimensional challenge that must take into account professionals, processes, technologies, and threat risk.

Slowing down developers isn't an option, so a new security model needs to be in place that's developer and DevOps friendly without giving up on security. Deep visibility enriched with full context and integration into developers' tools and processes, paired with unified cloud and container security, will drive better security practices.

Open source will play a significant role in cloud-native security. Proprietary security stacks will never match open-source security stacks. Black box solutions don't provide the required flexibility and transparency to meet the specific needs of organizations. With open source, organizations can fully understand and customize detection rules, as well as count on crowdsourced security policies, features, and bug fixes to better keep up with cyber threats.

The right approach to securing containers and cloud requires implementing security all the way from the source through production. This includes shifting left to protect the pipeline with image scanning and also managing configuration and access risks and detecting runtime threats and compliance violations.

Having founded Sysdig, one of the leading companies in cloud and container security, I understand how important it is to start from the basics when diving into new technologies. This book serves as a primer and simplifies complex topics to ease your path into learning about cloud security.

Happy reading.

Loris Degioanni

Founder and CTO at Sysdig

Introduction

Organizations are all at different stages of their cloud-native journey and want to take advantage of the speed, scalability, and automation of the cloud. From early lift-and-shift migrations to more mature application development using containers and Kubernetes, the cloud provides agility to ship applications faster.

This shift has introduced new security challenges, though. Security teams are now struggling to get visibility into their cloud environment, contextualize and prioritize investments, and deploy security solutions and processes without slowing down the business.

In the cloud shared responsibility model, cloud providers protect the underlying physical and virtual infrastructure that customers don't need to manage. But that means users don't have direct access to gain visibility and can't customize configurations. All services used to build and run applications and store data are the customer's responsibility to secure. In other words, the customer assumes responsibility for applying the necessary security controls and protection to manage security risk and meet compliance requirements for its cloud environments, including identities, permissions, and threat detection. So, cloud providers secure the cloud, and customers secure what they use, store, and run in it.

It can be tempting to try using existing security tools to protect cloud services. Unfortunately, these tools can't see everything happening inside clouds and containers to provide effective security. Using a tool designed for data centers also adds unnecessary complexity to workflows and slows down application delivery. Secure DevOps integrates security into the DevOps workflow and avoids slowing down software releases.

Vendors that are trying to ease adoption often tout an "agentless" approach to cloud security, but those rely on the limited information provided by cloud application programming interfaces (APIs). They fall short because these solutions can't see container runtime behavior, so they aren't suitable to detect the full range of cyber threats.

A better approach for cloud and container security is to adopt a unified approach that uses both agentless and agent instrumentation. Avoid using siloed tools because they create visibility gaps, inefficient processes, and excessive overhead. Tools that combine agentless and agent-based approaches offer the most comprehensive visibility and protection.

About This Book

This book gives you a look into better understanding cloud security and how you can adopt an approach that works for your company and secures all your containers and cloud environments. Secure DevOps is key to managing risk without slowing down application delivery. You need deep and unified visibility across your environments to see inside containers and see changes to cloud configurations and get context given the volume of microservices.

Icons Used in This Book

You may notice little pictures in the margins of this book. If you take the time to peruse them, here is what they mean:

The Tip icon gives you extra help, saves you time, or may even save you money. Check these out if you're looking for helpful hints on your security journey.

The Remember icon emphasizes important security facts. Keep these in mind when securing containers and clouds.

Pay close attention to Warnings. You want to avoid security threats and keep your environments running well, right? Then heed these warnings.

Sometimes we like to give you some background information or statistical information. When we do that, we use the Technical Stuff icon. You can enhance your security genius with these fun facts!

IN THIS CHAPTER

- Managing dynamic permissions and configurations
- Identifying cloud threats
- Having a unified threat detection approach

Chapter 1
Understanding Cloud Security

In this chapter, you discover how to manage cloud permissions and configurations, detect threats in the cloud, and apply a unified approach for cloud and container threat detection.

Managing Cloud Permissions and Configurations

As organizations mature in the cloud, the number of cloud services their teams use and the identity permissions that need to be managed grow exponentially. Many people refer to these services, which teams use to build and deliver applications, as *assets* or *resources.* Configuring cloud assets, roles, and permissions doesn't take long to become tedious, time consuming, and error prone.

WARNING

Misconfigurations of assets and overprivileged identities increase the risk of security incidents and are the leading causes of security incidents, so make sure that you diligently manage these.

IT organizations usually don't start with full control or even visibility over how their cloud assets are configured and which permissions were granted to identities in the cloud. Many cloud users manually configure cloud services as needed, like Simple Storage Service (S3) buckets, leaving IT out of the loop and potentially also leaving the company exposed to risky conditions. Other concerns include accounts of former employees, one-time users, and guest accounts that are left active, and also user identities with unused or unnecessary permissions.

Discovering cloud assets and configuration

Staying on top of existing cloud assets and their configurations can be challenging if done manually. IT sometimes misses the actual number by an order of magnitude. To keep up with the cloud's constant state of change, scale, and complexity, a programmatic approach is required. Manual processes, besides leaving blind spots, also increase the risk of missing an exposed asset with weak security controls configuration.

Misconfigurations could be the result of both unintentional (legitimate users) and malicious (attacker) actions. Regardless of the nature of the actor, paying attention to security posture by checking the status of dynamic cloud configurations is practically a requirement for cloud environments.


TIP

Cloud security posture management (CSPM) solutions offer cloud configuration management capabilities. For industry best practices, visit the Center for Internet Security (CIS) and review the CIS Benchmarks at `www.cisecurity.org/cis-benchmarks`. This resource provides you with checklists for all major clouds.

Identifying overprivileged users

Overprivileged entitlement of human and non-human identities (applications, services, containers, and so on) is a top cause of data breaches. Applying the *principle of least privilege* — the concept of providing no more permissions than necessary to perform required actions — is a wise, but difficult, concept to implement. Cloud providers make permissions granular, which in theory leads to least-privilege policies. However, the reality is much more complex.

In practice, permissions aren't assigned in a precise manner. Often, existing rules are reused, noting only if the permissions are broad enough to avoid disruption. In modern organizations, nothing should get in the way of speed and performance — not even security. So, developers and IT often err on the side of excess. Manual fine-tuning would be excessively time consuming and still not precise.

Inactive identities are also a permissions threat. These identities often get left off because organizations simply lose track and lack the ability to be alerted to this inactivity.


TIP

Cloud infrastructure entitlements management (CIEM) is a key tool to have in your cloud security toolbox. Look for solutions that can discover excessive permissions across active and inactive cloud identities and provide guided remediation to implement the least-privilege principle.

Detecting Threats in the Cloud

Threats can be seen as the activities of cybercriminals, such as phishing, data exfiltration, cryptomining, distributed denial of service (DDoS) attacks, and so on. Cloud threats today are elaborate and complex and have become completely out of the reach of traditional, siloed security solutions that use coarse, out-of-context, and non-real-time data. To detect and contain attacks effectively, you need real-time visibility of the full spectrum of malicious activities applied in the attack, including configuration changes that increase risk. For more information on today's global knowledge-base of adversaries' tactics, techniques, and procedures, visit `attack.mitre.org`.


TIP

The good news is that adversaries leave a trail of their actions in some form of recorded events. One way to detect threats in the cloud is to monitor cloud audit logs for anomalous activities and malicious actions, such as unexpected configuration changes and permission escalations.

Threat risk doesn't always result from malicious actions. Cloud configurations are changing constantly and must be monitored for impact to risk. When developers make configuration or

permission changes as they debug or deploy applications, they may not consider the additional risk this adds to the organization, so cloud and security teams must continually evaluate configurations against best practices and their organizations' policies.

WARNING

Some organizations send their logs to a security information and event management (SIEM) system and then scan for threats, but this approach has disadvantages:

- It's not real time, which imposes a delay in the detection of a risky configuration or an intruder.
- SIEM tools are more suitable for forensics analysis, not for real-time detection.
- Copying the huge volume of activity logs outside the cloud is costly and complex to manage and is a potential compliance violation in certain industries.

A more effective and efficient way to detect cloud activity threats is to apply stream detection. With stream detection, as the cloud audit records are generated, they're analyzed against defined runtime policies. If a suspicious action is detected, a security event is triggered in real time. Only the security event data is sent out, not all logged records. Also, each newly recorded log is analyzed against the conditions of the detection rules, not the entire audit logs storage.

With stream detection, you can detect in real time signs of cloud threats, such as the following examples:

- Turn-off of activity audit and logging services
- Change of user roles to add overpermissive policies
- Change to weak or no encryption of data at rest or in transit
- Change to weak password settings such as no multi-factor authentication (MFA) or no password rotation
- Change to inappropriate firewall rules or network access controls
- Creation of application programming interface (API) accounts with anonymous or unauthorized access
- Access to S3 buckets from unusual accounts

Adopting a Unified Threat Detection Approach

Today's universe of cyber threats is complex. Tampering with cloud permissions can just be a tactical step in an attack scenario that starts with the exploitation of a vulnerability in a workload, and being stealthy is the modus operandi. Adversaries adopt evasion techniques to get around legacy tools' defenses and also take advantage of visibility gaps left by siloed solutions.

REMEMBER

To detect and stop cyber threats in any environment, the first step is to *see* them. Trying to piece together data from siloed solutions slows down detection, and you may even miss the threat. If you don't see the threat, you can't stop it from spreading. As malicious activities could be happening in your applications, containers, Kubernetes, cloud assets, servers, and serverless platforms, a unified approach to threat detection is critical.

In this section, you discover the importance of unified threat detection for cloud and containers and why having a single event store, common policy language, and open source-based solution provides the foundation for unified security.

Single event store

In a unified approach, all detected events are in a single event store, enabling a sequenced event timeline that shows the attack in evolution. Siloed data slows down detection and may also be completely unsuitable to detect subtle and malicious aspects of a single action. When you use siloed security tools, you only see each activity in isolation, never putting together the complete attack. In sophisticated attacks where multiple systems are infected with malicious artifacts, like backdoors and malware files, without complete visibility of the attack components and blast radius, it could take months to completely remove the attacker from the environment.

REMEMBER

A centralized view, revealing attackers' sequence of steps from initial access through lateral movement and malicious actions across cloud and container environments, empowers security teams with the necessary information for immediate containment and removal of malicious artifacts.

Unified policy language

Detection policies should be set consistently across your cloud environment. Different policy languages add a learning curve and semantic gaps and may introduce parsing and translation loss when evaluating events. A single policy engine to detect threats across cloud and containers increases efficiency of security teams' workflows, which not only makes policy management easier but also reduces mean-time-to-recovery (MTTR) incidents.

Open-source validation

Security demands validation and transparency, so you want to avoid proprietary solutions. Proprietary tools are usually controlled by a single organization, and innovation is constrained by their resources and priorities. Solutions based on open-source standards usually have a primary contributing organization, supplemented by a community of motivated users and contributors that bring additional ideas and features. Open source also enables a more dynamic environment for providing feedback, reporting and fixing issues, and contributing with improvements. Technology ecosystems also grow much faster around open-source projects because having a community-based standard protects investments made in developing integrations. The synergies found in open source drive the collaboration, validation, and speed of innovation that are fundamental to combat modern cyber threats.

Just having security solutions in place isn't enough; you need the confidence that they're effective, can sustain protection against the full range of attacks, and continue to evolve to contain new threats at the same speed as the cybercrime world is evolving.

TIP

For cloud threat detection consider Falco. Falco, created by Sysdig, is the open-source standard for continuous risk and threat detection across Kubernetes, containers, and cloud. Falco acts as your security camera that continuously detects unexpected behavior, configuration changes, intrusions, and data theft in real time. Sysdig donated Falco to the Cloud Native Computing Foundation (CNCF), the vendor-neutral home for many open-source projects.

IN THIS CHAPTER

- Getting to know IaC security
- Looking for misconfigurations
- Finding drift
- Applying policy as code (PaC)
- Securing IaC with shift-left

Chapter 2
Securing Infrastructure as Code (IaC)

To keep up with modern demand for speed and scale, IT operations needed to change how they managed infrastructure. Infrastructure as Code (IaC) technologies were created as a new agile programmatic approach to infrastructure provisioning and maintenance.

IaC is incredibly powerful and enables transformative concepts like immutability, which means that changes are never applied directly to a workload in production; if a change is necessary, a new version is deployed and the old one deleted. With IaC, DevOps teams can provision infrastructure in seconds instead of days or weeks. If the templates are carefully constructed, the consistency reduces human error in configurations. However, in the same way that IaC scales and speeds up deployment, it can also expose the organization to critical security and compliance risks in seconds if security best practices aren't integrated up front.

In this chapter, you find out about securing IaC, what it means, and how it's implemented from source to production.

Understanding IaC Security

IaC allows engineers to provision servers with specific operating systems, run containers and Kubernetes clusters, configure storage and networks, and even integrate third-party services, all using template files. Before IaC, IT administrators, operation teams, and developers had to manually configure item by item of every stack layer deployed.

TIP

IaC delivers multiple benefits:

- Speed
- Scalability
- Resilience
- Cost reduction
- Addresses inconsistencies between development and production environments

But what about security?

Misconfigurations in the IaC template (such as Terraform, Amazon Web Services [AWS] CloudFormation, YAML files, Helm Charts, and so on) can easily expose the environment. Adversaries benefit from IaC misconfigurations that weaken the security controls of an organization and facilitate attacks. For instance, misconfigured Simple Storage Service (S3) buckets that leave sensitive data publicly accessible, missing access controls on cloud services application programming interfaces (APIs), risky configurations in Kubernetes deployments that facilitate container escape, lateral movement, and resource exhaustion, among other threats. This brings up the question — how do you implement IaC security?

Implementing IaC Security

Certainly, one must check for misconfiguration proactively before deploying into production but also detect changes while in production. For the required agility in scale, policies must be centrally managed and programmatically implemented. This section gives you the important aspects to consider.

Scanning for misconfigurations in IaC pre-production templates

At the volume in which infrastructure is created and modified in the cloud, manually configuring resources eventually becomes tedious, leading to errors and compliance violations. The bottom line is that IaC templates can't be left unchecked before deployments. Scanning for misconfigurations in IaC pre-production templates is a part of a shift-left approach in security, which means introducing checks and remediation during the development phase, and represents a fundamental step for a secure DevOps workflow.

TIP

As your developers increase the code delivery pace, make sure to put guardrails in place so they can maintain agility without compromising security. Create a baseline for security policies applied to IaC templates and check for violations before deployment in production. In today's agile environment, automation delivers speed and consistency, so make sure that IaC security checks are integrated into your continuous integration and continuous delivery (CI/CD) pipelines.

Detecting runtime drift and applying auto-remediation

Runtime drift is the difference between the originally defined values and what's running in production. Applied to IaC, a runtime drift means that the current configurations in production differ from the values defined in your policies and checked predeployment. For instance, is a S3 bucket now allowing simple password access?

Even well-implemented IaC security for cloud configuration templates in a CI/CD pipeline will suffer from drift. Drift can result from intruders' malicious actions and also from well-intentioned employees, such as site reliability engineers (SRE) acting on emergencies and DevOps teams applying a direct change to a cloud asset. The bottom line is that drift happens. Cloud security solutions must continuously check the state of cloud assets and services to verify if they match what has been defined in the respective deployment template.

Runtime drift can also happen in workloads, such as Kubernetes clusters. Just a few examples of drift in Kubernetes include

- The API server allowing anonymous authentication when it initially was set to deny
- Containers running as root when privileged containers aren't allowed
- Pods running containers without resource utilization limits set

Ideally, you shouldn't have to define pre-production and runtime security policies separately. The same IaC security policies applied in the CI/CD pipeline can and should be applied at runtime. This approach offers many advantages:

- It avoids increased overhead and gaps due to different policy language semantics and granularity.
- It enables easier automation of remediation.
- It can be seamlessly mapped back to the IaC configuration (template) file and then be fixed via a pull request (PR) and reconciliation rechecked across CI/CD pipeline and production.
- Auto-remediation with a simple Git PR to fix the violation at the source reduces exposure and mean-time-to-recovery (MTTR), improving the performance of the security, development, and operations teams.

Applying policy as code

As infrastructure becomes increasingly deployed as code, you must also define and manage security policies as code to keep up. Policy as code (PaC) allows you to leverage a shared policy model across multiple IaC, cloud, and Kubernetes environments. Not only does PaC provide consistency and strengthen security, but also it saves time and allows you to scale faster.

PaC can be implemented by using the open-source Open Policy Agent (OPA), which provides a framework for policy-based controls in cloud-native environments. For instance, organizations can create one OPA policy that can be consistently applied against different cloud and Kubernetes environments and across all stages of the application life cycle. OPA offers a common and powerful

language that allows security teams to decouple security policy management (for example, read, write, analyze, version control, and so on) from the management of the cloud infrastructure and services.

Implementing PaC minimizes risk by providing security teams with a way to centrally define and manage policies, removing opportunities for human error and internal threats. With a partnership between developers, IT operations, and security teams to define and deploy consistent policies, you lay a foundation for your organization to move fast with less cyber risk.

Securing IaC from source to production

The bad guys are taking advantage of cloud misconfigurations and learning how to exploit Kubernetes weaknesses. Identifying and eliminating risky configurations pre-deployment and in production must be a top priority.

IaC security starts with a shift-left approach and also enforces security and compliance at runtime, detecting and correcting drift from original values. By implementing a unified workflow that defines and enforces IaC security policies programmatically from source to production, organizations can improve productivity, close security gaps, and enable stronger collaboration between developers, operations, and security teams.

IN THIS CHAPTER

- Seeing the vulnerabilities in containers
- Using vulnerability scanning
- Integrating vulnerability management

Chapter 3
Preventing Vulnerabilities

Vulnerabilities are software bugs or weaknesses that could be used by an attacker. They could be present in the operating system, application code, and third-party code dependencies, such as libraries, frameworks, programming scripts, and so on. By taking a secure DevOps approach and identifying vulnerabilities early in development, you avoid frustrating developers with delays when an application is ready for production. So, preventing vulnerable workloads from entering production is paramount, but keep in mind that new vulnerabilities and exploits can be discovered for software already in production. Scanning for vulnerabilities must be done throughout the entire workload life cycle, including at runtime.

In this chapter, you discover how to identify vulnerabilities in container images and hosts, integrate vulnerability assessment and controls in your software development life cycle, and unify container and host vulnerability management.

Understanding Vulnerabilities in Containers

Modern cloud-native workloads run as microservices built on containers and deployed as Kubernetes clusters in the cloud. A vulnerability in a container can be exploited to grant attackers a foothold into your Kubernetes cluster, and from there, they can find ways to move laterally in your cloud environment.

According to the *Sysdig 2022 Cloud-Native Security and Usage Report*, 75 percent of images running in production contain patchable vulnerabilities of high or greater severity. The risk of exposing your organization through vulnerabilities present in containers is *very* real.

When securing containers, it's important to understand where vulnerabilities could be and how they could be introduced in your environment. Containers are built from read-only files called *container images.* Container images can be built totally from scratch or, more commonly, by using other available images, creating a layered structure. A base image (for example, Ubuntu, Debian, BusyBox, Alpine, and so one) provides the operating system (OS) dependencies and serves as a foundation for all other layers. The next layer could be a language or framework image, such as Ruby Gem, Python PiP, Node.js npm, and more. By using those readily available images, developers speed up development with a jump-start. But, they could also be inadvertently introducing vulnerabilities into their containers.

Many developers aren't aware that even official images stored in popular registries aren't safe. Unfortunately, these repositories have been used as a channel to disseminate vulnerable software by malicious actors, like in supply chain attacks. For instance, DockerHub repository is known for having high-severity or critical vulnerabilities in a significant portion of its publicly available images. Some repositories offer vulnerability scanning; however, the scanning is often done just on OS images.

According to the *Sysdig 2022 Cloud-Native Security and Usage Report*, a high rate of high-severity or critical vulnerabilities exist in non-OS image layers. Both OS and non-OS images must be scanned.

Image scanning is a must-have in container environments. The application code and all image layers' dependencies (OS and non-OS) in the container could have software vulnerabilities. As new vulnerabilities in existing software packages can be discovered at any time, scanning and auditing container images isn't a one-time deal. The process requires continuous monitoring that starts with your code development and extends to production.

Integrating Vulnerability Scanning into DevOps Life Cycle

Integrating vulnerability scanning into the software development life cycle (SDLC), as for Infrastructure as Code (IaC) security, starts with a shift-left approach by embedding scanning early in the continuous integration and continuous delivery (CI/CD) process. Checking container images earlier ensures that development can run fast without costly fixes to the final build, and security isn't a blocker for delivery.

This section lists some best practices to prevent vulnerabilities in container environments, including integration with Kubernetes admission controller as a last line of defense to block a vulnerable container from being deployed in a Kubernetes cluster.

Automate vulnerability scanning in the CI/CD pipeline

Integration with developers' existing tools is key to successfully introducing security into DevOps processes. So, you should embed image scanning into the CI/CD pipeline to catch vulnerabilities (in all image layers, dependencies, and artifacts) early in the development life cycle. By doing so, you will also be saving engineering cycles required in more complex late fixes. Integration with CI/CD tools, such as GitHub, Google Cloud Build, and Amazon Web Services (AWS) Codebuild, can be done in two main forms, by pulling out the images for scanning or via inline scanning.

Inline scanning offers a more secure process. With inline scanning, your application code doesn't leave your environment because you don't need to send a copy to an intermediary repository for scanning. Only the metadata of the scan results are sent out for

policy evaluation. Scan results are passed back to the pipeline, resulting in a pass or fail event.


TIP

Take a look into remediation cases that could be automated or facilitated, such as vulnerabilities with a fix requiring a low-risk version update of a package. And for those vulnerabilities without a fix or with a fix that would require more complex remediations, consider compensating controls with a risk assessment.

Automate registry image scanning

Keeping your private image repositories and registries free from vulnerable container images is another security best practice. Organizations commonly use registry services like DockerHub, Amazon Elastic Container Registry (ECR), Google Container Registry, and JFrog, among others. Even though these registries offer vulnerability scanning features, make sure that they're comprehensive. Make sure that your registry covers each and every image layer and OS and non-OS dependencies. Most only offer OS image scanning, so your scanning solution should check for vulnerabilities in non-OS packages as well. Developers may be unknowingly pulling in vulnerabilities from non-OS open-source packages such as Python PIP, Ruby Gem, and more, which introduces security risks.


REMEMBER

Protection against image tampering is also important. Make sure to digitally sign images and check signatures before using them. Verifying image signatures can assure that the specific image digest has been signed by the publisher.

As in the case of CI/CD pipeline integration, image scanning integration with registries can be done by pulling out the images to execute the scan or by scanning them inline. Organizations usually prefer inline scans that are done without pulling their images to a third-party repository for scanning.

Enforce vulnerability security with Kubernetes admission controller

Kubernetes admission controllers are the last line of defense before exposing the organization to container vulnerabilities in Kubernetes clusters. The admission controller evaluates requests to the Kubernetes application programming interface (API) server, then determines whether to allow the request. As an example, a

request for a Pod deployment could be denied if the admission controller receives a failed status on a check.

Admission controllers offer an extensible solution to enforce security policies using webhook integrations. Therefore, image scanners can be integrated with the admission controller to provide vulnerability check status prior to a container deployment in production. Kubernetes admission controllers also provide flexible rules to accept or reject requests based on the environment specifications. So, it is important that the vulnerability scanning solution also supports rich context within scan results so the environment context such as cloud, container, Kubernetes, and application criticality, among others, can be used to assess risk and define the appropriate policies.

Prevent vulnerabilities in serverless workloads

In serverless computing, cloud providers dynamically assign resources to run your workloads, but in a shared responsibility model, you're responsible for protecting these workloads. Vulnerabilities in your code and dependencies can still provide attackers with a way to compromise your environment. You can and should implement vulnerability scanning on serverless workloads as well.

For instance, before launching a task on AWS Fargate — a serverless computing platform service to run containers as a service (CaaS) without the need to manage the underlying infrastructure — you could automatically trigger the scanning of the container image in the AWS ECR repository. And again, scanning can be implemented inline or by pulling out the image. Make sure that all layers are verified for vulnerabilities and that you can block deployment if high-risk ones are found.

Manage vulnerabilities at runtime

New vulnerabilities and exploits are discovered every day on new and old versions of software packages and OS distributions. If you're only scanning at the build stage, you're leaving yourself exposed to recently discovered vulnerabilities.

Managing vulnerabilities in your runtime environment is necessary. Your vulnerability management solution should keep track of images deployed in production and alert when an image is

affected by new vulnerabilities in feed updates from vulnerability data sources. Security tools use different strategies to achieve this. One way is to rescan all images periodically, but that's not an efficient use of your resources. Ideally, the image metadata of deployed containers is stored and new vulnerabilities can be detected without the need of a full rescan.

Let us just add this additional observation point: Alerting to vulnerabilities at runtime without context isn't really effective. Environment information, such as cloud, Kubernetes, and application context, is necessary for risk assessment, remediation prioritization, and guidance regarding containment and compensating measures.

Implement host scanning

Hosts are the computing instances where your containers run. In Kubernetes environments, hosts are the nodes of a cluster where Pods are deployed, and an estimated quarter of organizations in the cloud have unpatched hosts with high severity and critical vulnerabilities. Although not as numerous as containers, vulnerable computing instances in public clouds expose the organization to serious risks.

Securing host virtual machines (VMs) is just as important as securing the containers running on them. Make sure that you have a tool in place that can scan hosts. For example, cloud VMs (computing instances such as EC2s) are hosts that need scanning. Host scans must also be done regularly, so your teams are given actionable information to prioritize and expedite remediation.

Adopting a Single Vulnerability Management Workflow for Containers and Hosts

Siloed solutions create security gaps and inefficiencies. Adopting a unified approach to vulnerability management for containers and hosts speeds the time to detect and remediate vulnerabilities, while generating fewer alerts. Organizations are often faced with thousands of vulnerabilities detected in their environments while their teams can only timely handle a small fraction of them.

Prioritization and efficient remediation have become fundamental to vulnerability management.

REMEMBER

Risk comes in different forms. Not only do vulnerabilities in the code expose your environment to threats, but also it's important to verify if security best practices are followed as well as all compliance standards and regulations that your organization adheres to.

In this section, you dig deeper into vulnerability prioritization.

Implement vulnerability prioritization

How to prioritize which vulnerabilities to fix is a constant challenge for resource-constrained security teams. Evaluating only the severity of a vulnerability may lead to cycles spent on vulnerabilities that don't represent practical risk. When you're doing prioritization, the crucial decision to make is about what can't wait. So make sure to also include in your analysis contextual risk that reveals the true relevance of a critical vulnerability to your environments.

Think about the following questions:

- Is the vulnerability located in packages loaded when the container is running, and therefore with real chances of being exploited, or in a package that isn't used?
- Is the vulnerability exposing critical applications?
- Is the vulnerability present in environments that aren't externally accessible?
- Are there available exploits being used, and if so, what's the reported threat activity?

As you answer these questions, it will become clear which vulnerabilities in your environment context are truly exploitable and incur high risk to the organization. Those areas are the ones where you need to focus your remediation efforts.

Managing vulnerabilities requires understanding how to effectively reduce risk in your environment. Make sure that your vulnerability scanning solution provides detection of vulnerabilities on containers *and* hosts. Check the quality of the threat data and context provided with the vulnerability alert for prioritization and remediation, tracking improvements in reducing exposure risk.

Check for compliance requirements

Unified host and container scans can be used beyond vulnerabilities checks. The scans can also validate regulatory compliance (for example, PCI DSS, NIST regulations, SOC 2) and security best practices, such as the Center for Internet Security (CIS) Benchmarks. Scanners should detect security risks, such as port misconfigurations, unprotected secrets, open-source software (OSS) licenses, and file integrity, among security controls.

For more information on regulatory compliance obligations, check out Chapter 5.

IN THIS CHAPTER

- Understanding runtime security
- Using Kubernetes NetworkPolicy
- Being prepared for incident response

Chapter 4
Detecting and Responding to Threats

Security is top of mind for cloud security and DevOps teams. Shifting security left by removing risky configurations and vulnerabilities before deployment into production does the important job of hardening your environment. However, according to the *Sysdig 2022 Cloud-Native Security and Usage Report*, 75 percent of containers running in production contain high or critical vulnerabilities. In practice, most organizations don't fix all vulnerabilities, so they need a safety net. And, of course, there will always be zero-day exploits and internal threats. Detecting and responding quickly to suspicious behavior is critical. The sooner an attack is discovered, the faster you can stop it.

In this chapter, you find out about runtime security and how to detect threats in your environments, apply Kubernetes network policies to block suspicious traffic, and enable compliance audits and incident forensics.

Defining Runtime Security

Runtime security is all about detecting anomalous behavior and threats in production. In cloud-native environments, runtime security is done by observing and alerting to unexpected suspicious activities in the containers, Kubernetes, and cloud services.

Detecting threats on containers and hosts

Containers should be monitored at runtime to detect unexpected behaviors related to processes, file input/output (I/O), network traffic relationships, and user activities. Profiling container behavior and using the baseline to create detection rules is a good starting point. But every organization is unique, so make sure that detection policy rules can be easily adjusted to accommodate exceptions in your environment. This will improve accuracy, reducing noise.

Attackers can also target the host on which the containers run. For instance, in a container escape, attackers cloud exploit host vulnerabilities, deploy malicious code to launch campaign attacks, try lateral movement, and compromise your whole infrastructure. Infected hosts can be used by command-and-control (C&C or C2) servers for various purposes, including cryptomining, distributed denial of service (DDoS) attacks, ransomware, data exfiltration, espionage, and so on.

Being able to detect malicious and anomalous behaviors on containers and hosts is critical to protect your environment against threats. However, having two different solutions for hosts and containers increases overhead, creates detection gaps, and investigation burden. A unified approach with a common policy language and source of truth is ideal. Deep visibility into system calls (syscalls) to the operating system (OS) kernel provides a reliable source of truth for event data across hosts and containers. Syscalls visibility can be provided by collecting kernel events via kernel modules or eBPF probes.

USING FALCO FOR CLOUD-NATIVE RUNTIME SECURITY

You can use Falco open source to monitor system calls and apply detection rules to alert on anomalous behavior. Falco can detect, in real time, when suspicious process, file, network, and user activities occur on containers and hosts and has become the de facto standard for cloud-native runtime security.

Kernel modules are pieces of code loaded into the kernel to extend its functionality. For example, kernel modules can be used to intercept and copy details of system calls to provide deep visibility into system activity. *Extended Berkeley Packet Filters* (eBPFs) are Linux-native technology that enables the safe execution of programs in the kernel. eBPF probes can also be used for observability of system call events and are an alternative to kernel modules. eBPF offers the additional advantage of already being part of the LINUX operating system, and therefore, it's immutable infrastructure-friendly.

Detecting threats in Kubernetes clusters

Kubernetes has become the "operating system" of the cloud. Its orchestration capabilities enable fast, resilient, and scalable containerized production environments. But Kubernetes can be difficult to secure. Visibility at runtime in Kubernetes is a challenge and a source of concern for security and operations teams because they're often flying blind.

Traditional tools, such an endpoint detection and response (EDR), can't provide visibility into dynamic cloud-native container environments. How can you secure Kubernetes if your detection mechanisms don't see what's going on in containers or what's happening in the cluster? Securing Kubernetes requires a secure DevOps approach that combines deep visibility into system events and activity in the orchestration layer.

DEEP VISIBILITY WITH FALCO

Falco provides the deep visibility required to secure Kubernetes at runtime. Falco policy rules can look at anomalous behavior inside containers and also detect suspicious user activity analyzing the stream of Kubernetes audit logs. A suspicious process running that's followed by a Kubernetes application programming interface (API) call can reveal a compromise.

The open-source community has been creating Kubernetes security rules with Falco, so you don't have to start from scratch. Out-of-the-box policies map to security frameworks, such as MITRE ATT&CK, to help security operations center (SOC) teams streamline their detection and response workflows.

Detecting threats in the cloud

Another important aspect of runtime security is to check for suspicious activity or risky behavior being performed in the cloud. Cybercriminals will leave an activity trail that gets recorded in cloud activity logs when carrying out their attacks. Compromised cloud accounts will have intruders changing permissions, deactivating multi-factor authentication (MFA), and leaving buckets exposed to the public, among other malicious activities.

FALCO IN THE CLOUD

Monitoring cloud audit logs, such as Amazon Web Services (AWS) CloudTrail, Google Cloud Platform (GCP) Audit Logs, and Azure API Management logging records, is how you can detect malicious actions in the cloud. And Falco is the tool for the job. Because Falco analyzes cloud audit logs against security policies using stream detection, instead of looking for patterns in a large log store — as is done with security information and event management (SIEM) solutions — it can catch an intruder in real time.

Falco can also improve the cost-effectiveness of SIEM solutions. Exporting data outside the cloud is expensive. Using Falco's stream detection to send only detected events to SIEM saves money by reduced data transfer expenses and SIEM data storage fees. By reducing incident response time, you reduce the risk of data loss.

Blocking Unexpected Traffic with Kubernetes NetworkPolicy

By default, all pods within a Kubernetes cluster can communicate with each other without any restrictions. In the case of a container compromise, the whole cluster is at risk of a threat moving laterally. Segmenting network access by giving permission to only what's needed is recommended because it would limit the blast radius of a breach. Kubernetes NetworkPolicy resource offers a mechanism to do exactly that.

TIP

Kubernetes NetworkPolicy resource doesn't require deploying a sidecar (an additional container deployed in the same pod with the application container) proxy, so you can enforce network security without additional operational burden.

One way to implement network policies is to profile your containers' network traffic and create a baseline for the communications expected to and from the containers. For instance, what other services does the container communicate with? What are the ports used? Does it handle traffic from the internet?

After the container network traffic behavior is well understood, it can be used to define the ingress and egress policies in Kubernetes NetworkPolicy resource. All traffic that differs from what's defined as allowed will be blocked. By using network policies in your Kubernetes cluster, you can implement effective network segmentation aligned to Zero-Trust principles of microsegmentation.

Planning for Incident Response and Forensics

Container environments also create challenges for incident responders and security forensics teams. Containers are ephemeral, with many lasting for seconds or only a few minutes; after they're gone, how do you investigate them? Matters get worse in "container as a service" environments that offer no visibility or control of the underlying hosts and runtime environment. For instance, if a workload task performs a suspicious file system

change, how do you verify if there was a breach and a compliance violation? If containers are compromised, attackers can use them for cryptomining or perform lateral movement to gain full access to your data.

Detecting, responding, and investigating threats in containers running on host instances or on serverless services requires deep visibility into workload activity. So, make sure that you can collect granular data on workload activity and an audit trail to investigate what, who, and how it happened. The volume of cloud security events is growing and having the necessary data to rapidly respond to incidents on hosts and containers is critical.

Get deep visibility before and after an incident

A source of truth, such as syscalls, that reveals process, disk, and network activity before, during, and after the event will expedite investigation. Visibility into user actions and commands used will also help connect the dots faster. Make sure that your security solution collects data for forensics for a postmortem analysis, so you're able to answer the "when," "what," "who," and "how" questions that come up during an investigation.

Get full Kubernetes and cloud context

TIP

Make sure you leverage rich cloud and Kubernetes context metadata to better prioritize incidents. Lack of context will result in alert overload and higher mean-time-to-recovery (MTTR) in cloud-native environments. Also, look for solutions that present all events in a unified timeline. It will make it easier to understand a potential cyberattack chain with a sequence of events across the cloud and containers. SOC and DevOps teams should have security and contextual information readily available.

Integration of detection, investigation, and remediation workflows will also improve MTTR. Having quick access to suspicious containers for local investigation and mitigation straight from the alert can save many cycles of investigation. Looking directly into local logs, container files, and process/memory dumps will reveal what adversaries may be trying to hide. With full stack context and a clear picture of the extent of the incident, faster, more effective, and more efficient remediation can be done.

IN THIS CHAPTER

- Looking at the challenges of compliance
- Applying compliance checks
- Taking advantage of open-source tools

Chapter 5
Keeping Containers and Cloud Compliant

From internal security practices to industry regulations and even laws, all organizations are subject to some kind of cybersecurity or regulatory compliance obligations. Specific markets and industries have specific regulations and standards they enforce.

Guidelines also come in the form of best practices, but regardless of the type of enforcement applied, meeting compliance is a must have for organizations moving to the cloud. Check out the later section "Implementing Compliance Checks Across Cloud and Containers" for more information on common regulatory standards and cybersecurity best practices.

In this chapter, we tell you all about the challenges of cloud compliance, how to check compliance and governance, and what open-source tools you could use.

Facing Compliance and Governance Challenges in the Cloud

Cloud environments' configuration flexibility and dynamic nature make it hard to meet compliance, stay compliant, and prove it. Ad hoc compliance assessments don't work for the cloud. Ever-evolving and increasing threats augment the risk of non-compliance and demand continuous checks.

Cloud and Kubernetes security controls are foundational to compliance standards and regulations. Unfortunately, misconfigured security controls are common due to user errors and malicious actions. Staying compliant requires a continued effort in cloud security posture management (CSPM) and Kubernetes security posture management (KSPM). Compliance requires that security posture must be enforced at provision, checked for drift in production, and validated via audit reports.

REMEMBER

To achieve and manage compliance, your organization needs to take critical action in the following areas:

- **Asset and configuration inventory:** The first challenge faced in achieving and managing compliance is keeping an up-to-date cloud inventory. From cloud services to Kubernetes and serverless containers, visibility is fundamental for compliance. Make sure to discover all cloud assets and their respective configuration.
- **Mapping of controls:** After you assess your inventory, map relevant compliance standards and regulations to cloud and Kubernetes configuration parameters. Tackling this area means getting all control points aligned to satisfy the requirements of compliance, security, and audit teams.
- **Policy enforcement:** Create policies to check and enforce the controls (from the preceding bullet) in an automated fashion. Applying consistent policies across environments facilitates automation of compliance and governance.
- **Validation:** On-demand assessments are core to passing audits as well. Providing satisfactory evidence in audit processes requires periodic and on-demand checks accompanied by comprehensive reports that show proof of compliance.

TIP

Detailed cloud and Kubernetes audit logs, as well as granular container forensic data, such as system calls, are good sources for evidence that prove compliance.

Implementing Compliance Checks Across Cloud and Containers

Implementing regulatory and corporate compliance checks for cloud and container environments, such as Kubernetes, can be overwhelming, but it doesn't have to be. Some commercial security products have mapped requirements for standard compliance controls to policy rules and can detect violations.

TIP

When you're implementing regulatory and corporate compliance checks within your organization, look for tools that offer out-of-the-box checks to meet common regulatory standards and cybersecurity best practices, such as the following:

- NIST 800-53
- SOC 2
- PCI DSS
- HIPAA
- GDPR
- The Center for Internet Security (CIS) benchmarks

FILE INTEGRITY MONITORING

File integrity monitoring (FIM) is a key capability to implement to check multiple boxes. It's a requirement to meet many standards and regulatory requirements. Additionally, because FIM gives you visibility into changes applied to your files and directories, it could reveal suspicious activities, such as access to sensitive data, tampering of container images, modification to critical system files, and so on. Therefore, threat detection in containers and hosts is also a core FIM use case. By monitoring file integrity, you can detect leaked credentials and installation of malware, for instance.

REMEMBER

Organizations can drift in and out of compliance between audits. Providing evidence that proves compliance is critical. Keep records of all container, Kubernetes, and cloud activity with rich context for fast interpretation. Those records are crucial to removing doubts of compliance when you're investigating incidents.

Compliance is continuous, requiring repetitive tasks. Therefore, organizations can benefit enormously from automation. From policies to detection of drift and reporting on compliance status, make sure that you look for solutions that enable automation of workflows. And, although compliance isn't the same as security, it should be part of a holistic cloud security strategy.

Using Open-Source Tools to Help with Compliance

Open-source tools are resources that help you meet your compliance and security goals. Open Policy Agent (OPA) and Falco are two open-source tools that can be used to manage compliance in the cloud, Kubernetes, and containers.

OPA

OPA provides policy-based controls that are flexible and fine-grained to conform with administrative needs in cloud-native environments. It provides a single policy language and application programming interfaces (APIs) to enforce compliance consistently in all stages of cloud resources provisioning, workload deployment, and runtime. For more information, visit `www.openpolicyagent.org`.

Falco

Falco open source is the most adopted tool for cloud-native runtime security. Its flexible rules engine allows the detection of practically any type of anomalous host or container activity, including compliance violations. To find out more, check out `falco.org`.

IN THIS CHAPTER

- Getting the most out of your application performance and availability
- Targeting issues quickly with granular metrics
- Simplifying your monitoring environments
- Steering clear of vendor lock-in

Chapter 6
Targeting Monitoring and Troubleshooting Issues

Over the past few years, the infrastructure and ecosystem evolution with microservices and containers has made many traditional monitoring tools and techniques irrelevant. Instead, developers need solutions that can adapt to the short-lived and isolated nature of containers and application services. As environments become more complex, simplicity and compatibility with open standards will be required to keep ahead of performance issues that can arise.

In this chapter, you dive into the key fundamentals of monitoring container environments.

Maximizing Application Performance and Availability

Containerized applications running in production must be constantly monitored for availability, errors, and service response times. Achieving this level of monitoring requires the collection of a wide range of telemetry and event data. Sources and approaches for collecting and presenting this information can vary.

REMEMBER

Measuring the internal state of your system through the examination of data outputs is called *observability.* In order to achieve observability when monitoring your application environment, you need to collect metrics, traces, and logs. We cover those in this section.

REMEMBER

After you've collected these data points, make sure to analyze them for anomalies that indicate the relative performance of your environment.

Metrics

As the foundation of monitoring, *metrics* are predictable counts or measurements that are gathered over a specific period of time. By collecting and correlating container metrics with infrastructure and orchestration data, you can monitor the performance, health, and state of your containerized applications and then maximize availability. An example of a metric is the amount of central processing unit (CPU) used on a system at a given point in time.

Logs

Logs are time-stamped records of discrete events that can store outputs from applications, systems, and services. They provide insight into what changes in your system's behavior when something goes wrong. For example, an application log may tell you what errors were registered just before an application crashed.

Traces

Traces are pieces of information about a transaction or request from your applications or systems that enable you to follow the transaction or request from beginning to end. A single trace displays the operation as it moves from one node to another in a distributed system. For example, traces help pinpoint where failure occurs and what causes poor performance of a purchase transaction in a web application.

Troubleshooting Problems Faster with Granular Metrics

A key challenge with troubleshooting containers is that they may no longer exist after a problem occurs. In fact, the *Sysdig 2022 Cloud-Native Security and Usage Report* states that about 44 percent of containers live less than 5 minutes. In some cases, this is by design.

Containers are often described as ephemeral constructs that last for only as long as they're needed. When their single task is complete, the container is stopped, destroyed, or otherwise disposed of while the application moves on to the next step in its process. Orchestration tools schedule and reschedule containers as the environment changes. Comprehensive container monitoring solutions should include the ability to automatically record all the activity on a system that takes place surrounding an event, like a container that was killed. Capturing information such as commands, process details, network activity, and file system activity allows after-the-fact investigation, even after containers are gone and outside the production environment.

REMEMBER

The important metrics and techniques to use when troubleshooting container environments include

- **Kubernetes resource utilization:** Make sure your monitoring tool can help you troubleshoot issues such as crashloop backoffs, pod evictions, and resource allocation and limits. See which containers are running out of CPU, memory, or file system resources leading to throttled or killed containers.
- **Event correlation:** Contextualize metrics and understand the "why" faster with a unified view of both metrics and events. Events from Kubernetes such as deployments and node failures, as well as alert events, security violations, and custom events, should be presented alongside your key metrics for faster correlation and problem resolution.
- **Application and service golden signals:** Golden signals can help you understand which signals are most significant in your environment. Golden signals for monitoring include latency, errors, traffic, and saturation. Understanding these signals can help make sure your application environment is

performing optimally. Set up dashboards that provide immediate visibility into what matters most for faster troubleshooting.

- **Analysis of syscall capture files:** In order to lower mean-time-to-recovery (MTTR), you should capture in-depth system call activity to more effectively triage issues in your container environment even if containers are already killed. This helps you reproduce step-by-step run processes and commands, reconstruct read and written files including logs, visualize traffic on network connections, and more. By having these details at your fingertips, you can find the source of the problem and resolve it quicker.
- **Segmentation of key metrics to focus investigation:** Don't rely solely on container level usage information. You need to get resource usage information at the process level inside the container to effectively troubleshoot problems. Segmenting key metrics by process can help eliminate false positives when troubleshooting container errors and performance problems. If you don't have container process level metrics, you won't be able to pinpoint the root cause.

Simplifying Complex Monitoring Environments

When you have complicated environments, like Kubernetes and containers, the last thing you need is a complicated monitoring tool with complicated procedures. A streamlined monitoring approach that still gives you the details when you need them is necessary to run cloud-native workloads efficiently.

TIP

In order to maximize your results, invest in the following:

- **Scopes:** Finding the source of a problem in a large Kubernetes environment isn't just like finding a needle in a haystack. It can feel more like finding a needle in a *needle stack.* With so many entities to examine, help yourself out and scope your monitoring metrics down to just a particular region, deployment, namespace, or workload to minimize the noise and focus on the signal you're looking for.

- **Pre-built dashboards:** Building meaningful dashboards can take a lot of time and effort, and you may still miss critical information if you aren't already an expert who knows what to look for. Find a tool that provides out-of-the-box dashboards designed to provide best-practice views of modern applications and infrastructure. Community resources also exist for dashboard templates that you can use to monitor popular applications and workloads, so you can immediately gain visibility into application health when monitoring Kubernetes and cloud services.
- **Pre-built alerts:** Effective alerting is at the bedrock of a monitoring strategy. Naturally, with the shift to orchestrated container environments and Kubernetes, your alerting strategy will need to evolve as well. Your alerts should be ready for you to enable with recommendations based on best practices, so you don't need to guess at the right thresholds; instead, you can tweak them to suit your needs. Use your monitoring tool to identify your golden signals and then set service level indicators (SLIs) and service level objectives (SLOs). These help you narrow down where and when your alerts need to be triggered.
- **Integrations:** The number of different applications and services you need to monitor in a Kubernetes environment is enormous. Finding a tool with a wide range of integrations for applications and cloud services eliminates blind spots and saves you time by eliminating DIY approaches and costly maintenance.

Avoiding Vendor Lock-In

Open-source software has become the way to ensure that applications are reliable, extensible, and secure. Unless there is a game-changing feature that can't be implemented in an open way, proprietary tools should be avoided because they lock you into their closed ecosystem, limiting your choices. However, the maintenance burden of using open-source software yourself can be costly. Often, companies would rather have employees spending time on business building efforts instead of maintaining tools. This is where open standards come into play. By buying and using tools that support open-source software and employ open standards, organizations can get the best of both worlds.

Key functionalities that benefit from adopting open-source standards include the following:

- **Metric collection:** Many monitoring tool vendors have deployed a proprietary approach to collecting and storing the metrics they collect. This means that in order to access those metrics, you're required to use their software. Other vendors use open standards for metric collection and storage, like the Prometheus project. This approach provides the maximum flexibility because you can use standard application programming interfaces (APIs) to interact with your metrics without necessarily having to install your own Prometheus environment.
- **Integrations:** Integrations for application and cloud services are necessary if you want to be able to correlate those metrics with the ones you're collecting from Kubernetes and other infrastructure resources. However, configuring and maintaining these integrations yourself can be time consuming. At the same time, using a vendor's proprietary integrations could leave you with reduced functionality and lead to vendor lock-in.

  
  TIP

  A better approach would be to pick a monitoring tool vendor that manages and supports the integrations for you while also using open source. An example of this would be Prometheus exporters. If your vendor can deploy and support Prometheus exporters as part of its monitoring software, you'll eliminate most of the maintenance and still be following open standards.
- **Querying your monitoring data:** Consider how you'll query your metrics to create meaningful views of your data. Pre-built dashboards can only go so far, and you ultimately need to create some custom views that address your particular use case or environment. Many vendors have created proprietary methods and query languages in order for customers to query the data in their proprietary metric store. This can again lead to vendor lock-in and limited functionality.

  
  TIP

  Look for a monitoring tool that uses a query language based on open standards for querying data, such as SQL or PromQL, to make sure you have the functionality and flexibility you need to create meaningful dashboards and alerts.

IN THIS CHAPTER

- Securing the CI/CD build
- Implementing cloud and runtime security
- Avoiding vendor lock-in

Chapter 7
Ten Considerations for Securing Cloud and Containers

Throughout this book, we present principles, processes, and technologies for organizations to run secure, compliant, reliable, and cost-effective cloud and container environments. A vast set of security controls, compliance requirements, and best practices need to be in place to achieve these goals. A good way to define your implementation strategy is in terms of specific objectives you want to meet on your journey. Most organizations adopt the cloud and containers to accelerate application development, and by adopting a secure DevOps approach and embedding security into the DevOps workflow, you can ensure security controls don't slow down developers.

In this chapter, you find key considerations to keep in mind as you put together your plan for securing clouds and containers.

Secure the CI/CD Build

Shift-left security integrates image scanning into the continuous integration and continuous delivery (CI/CD) pipeline. Make sure to scan both operating system (OS) and non-OS packages for vulnerabilities in every stage of the CI/CD pipeline to avoid costly fixes at deployment. Security best practices and compliance should also be checked by scanners, and it's a best practice to adopt inline scanning to avoid sending images outside of your environment. For more information on the CI/CD pipeline, flip back to Chapter 3.

Take Advantage of Kubernetes Native Controls

Prevent containers with vulnerabilities and risky configurations from being deployed into production clusters by using Kubernetes admission controllers. Kubernetes admission controllers offer flexible policy setting, and, as a native construct in Kubernetes framework, it offers a powerful mechanism to control what gets deployed in Kubernetes clusters.

Secure IaC Templates

You can shift security further left by implementing Infrastructure as Code (IaC) security. Enforce security policies in IaC by scanning cloud and Kubernetes templates for misconfigurations and violations of security best practices. Make sure you can detect drifts at runtime and automate remediation at the source with a pull request (PR). Visit Chapter 2 for more information on IaC.

Manage Cloud Configurations and Permissions

Ensure you have full visibility into cloud assets, identifying misconfigurations and drift across multi-cloud environments. Implement the least-privilege principle by detecting and removing

excessive permissions on user roles (human and non-human). Look for tools that can not only automatically discover all identity and access management (IAM) roles and their permission configurations but also can detect roles with overpermissions and recommend the right permission settings.

Implement Cloud Security Monitoring

Keep track of cloud assets and their configurations. Cloud misconfigurations can easily happen and are a leading cause of security incidents. Make sure that cloud log auditing is enabled for all services and monitored for threat detection. Cloud logs are important forensics records as well. Every cloud provider offers activity audit logs that show who did what and when.

Implement Runtime Security

Act fast on early indicators of compromise. Runtime threats are real and growing in sophistication. Adversaries are launching complex attacks to evade detection while infecting systems for maximum gain. Don't miss weak signals. Get real-time, deep visibility into events to detect suspicious behavior and malicious activity in the cloud, container, and Kubernetes. Check out Chapter 4 for more information on runtime security.

Enforce Zero-Trust Network Segmentation

Adhere to Zero-Trust principles by applying network segmentation and allowing only required communication between container services. Make sure that all network communications between pods, services, and applications inside Kubernetes follow network policies. Defining network segmentation manually is time consuming and error prone. Look for ways to automate.

Monitor Container and Kubernetes Performance and Availability

Monitor resource consumption and application golden signals to stay ahead of performance, availability, and capacity issues in your Kubernetes clusters. Cloud-native environments are complex, so make sure that the monitoring is simplified with scoping capabilities to focus on a particular region, deployment, namespace, or workload. Also, look for out-of-the-box dashboards and alerts, as well as easy integration with other data sources.

Have an Incident Response Framework for Containers

Implement incident response and effective forensic investigation processes to understand security breaches, meet compliance requirements, and recover quickly in cloud-native environments. Make sure that a source of truth is available for providing deep visibility into system calls, as well as all activity in the container and orchestration layers. Incidents don't happen in a vacuum; granular data must be available to reconstruct the attack before and after the incident.

Use Open-Source Tools to Avoid Vendor Lock-In

Embrace solutions based on open source to avoid vendor lock-in and take advantage of ecosystem integrations contributed by the community. Products based on open-source standards provide transparency and flexibility, so you can understand how detection rules are defined and customize to meet your needs.

REMEMBER

By adhering to standards set by the community, you can protect your investment in technology and more easily find team members with the skills you need.

Flip back to Chapter 6 for a more in-depth discussion about avoiding vendor lock-in.

Secure your Cloud from Source to Run

Get a single view of risk
across containers, Kubernetes
and cloud services

No blind spots
No guesswork
No black boxes

Secure your cloud from source to run

From early deployments to modern cloud-native environments, organizations struggle to get the visibility required to manage security risk and meet compliance in the cloud. Legacy solutions aren't effective; a new approach is required. This book provides key fundamentals to run containers, Kubernetes, and cloud with confidence by implementing secure DevOps. You discover how to find overpermission and misconfiguration, prioritize vulnerabilities from source to production, and stop modern threats.

Inside...

- Understand cloud and container security
- Manage permissions and configurations
- Secure Infrastructure as Code
- Prioritize vulnerabilities
- Detect and respond to threats
- Keep containers and cloud compliant
- Monitor and troubleshoot issues

sysdig

Go to Dummies.com™
for videos, step-by-step photos, how-to articles, or to shop!

ISBN: 978-1-119-84345-0
Not For Resale

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.