RAPID7

WHITEPAPER

Securing Kubernetes With InsightCloudSec

Rapid7's Cloud-Native Security Platform

TABLE OF CONTENTS

Kubernetes: The Innovation Engine for Cloud-Native Applications

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation. The name Kubernetes originates from Greek, meaning helmsman or pilot. Kubernetes software was developed by Google and open sourced in 2014. Since its creation, Kubernetes has benefited companies by improving resource allocations and shortened software development cycles.

Kubernetes is the natural evolution of developers' needs to manage resources for multiple applications running in a shared space. Back when applications were all running on an organization's servers, resource sharing was often an issue - if one application monopolized resources then it would starve others. VMs partially alleviated the problem by creating self-contained, isolated environments that prevented applications from competing directly with each other for resources. VMs could provide an environment for applications to run on that might even present multiple physical machines as a single system, enabling excellent scaling. Virtualization also provided a measure of security, as applications in one VM would not be able to access the data in another VM. However, because VMs required all the components of a full machine (including the operating system on top of the virtualized hardware), they created overhead challenges.

Containers extended the concept of virtual machines by sharing common resources like the OS and making them more lightweight in comparison to VMs. While a container has its own resources - including a filesystem, share of compute resources, memory, and process space - they can be run on the cloud and wherever a compatible operating system is offered, as they are decoupled from the underlying infrastructure. What containers lacked, however, was a way to ensure they are running properly with zero downtime. If a container goes down, a system is needed to immediately start a replacement container.

Kubernetes provides a resilient and reliable way to manage containers and distributed systems. Kubernetes manages deployment, scaling, and failover for applications, with tools for load balancing, storage orchestration, automatic bin packing, automated restarts of containers, and Secrets management.

Kubernetes does more than just provide management and orchestration tools for containers. It enables a software development process with multiple development teams working in parallel to develop, test, and deploy software rapidly and efficiently. Instead of one major software release every year, enterprises using Kubernetes can release improvements to their applications weekly, daily, or even multiple times a day, enabling developers to be agile and responsive like never before, and on a global scale.

Game-Changing Enterprise Cloud Security

Enterprises must clearly understand the two major structural changes that Kubernetes brings to software development to effectively secure the development process.

1. **Kubernetes applications are not limited to static server clusters or even virtualized environments.** Kubernetes enables lightweight application containers to be deployed on any compatible operating system and infrastructure. Kubernetes-based applications are therefore extremely elastic and can scale and self-repair faster and more cost effectively than with older methods of running applications.

Traditional cloud security methods assume that the security perimeter is definable. A company's perimeter might be a server, data center, or a network of servers, data centers, and cloud providers. Security teams depend on this ability to define borders to monitor and secure S3 buckets, Azure instances, and remote access. But with Kubernetes, containers might be deployed anywhere and may interact with each other in unknown ways. Even when container interactions are carefully mapped, hackers are constantly looking for ways to infiltrate and exploit vulnerabilities, including the nightmare scenario of container escape, where a container is taken over by an intruder and used to access private information or run unauthorized functions, such as cryptomining.

Key takeaway: Security teams must understand how Kubernetes-based applications function and employ security measures to mitigate the novel risks introduced by Kubernetes-native applications.

2. **The automated rollout capabilities of Kubernetes enables developers to deploy changes to containers in a controlled manner.** Development teams can now structure their work in a new way and work in smaller, parallel teams to deploy changes independently of others. This move away from monolithic software engineering enables rapid development and deployment of applications and upgrades, so companies can be more agile and responsive.

The rapid development, testing, and deployment enabled by Kubernetes introduces risks that may be structural or process based. For example, developers seeking to quickly test functionality may code passwords in the open or bypass critical security configurations. An analysis conducted by Alcide in 2019[1] showed that 89% of Kubernetes deployments coded sensitive passwords in the open and did not use the native Secrets management functionality offered in Kubernetes. In another example, developers may turn off security restrictions and forget to turn them back on before moving code to production, a vulnerability-creating process called security drift.

Key takeaway: Security teams must understand the Kubernetes software development lifecycle to proactively identify and fix areas where vulnerabilities might be introduced and where security drifts might occur.

[1] Alcide: New Analysis by Alcide Finds 89% of Kubernetes Deployments Not Leveraging Secrets Resources: http://www.globenewswire.com/news-release/2019/07/15/1882828/0/en/New-Analysis-by-Alcide-Finds-89-of-Kubernetes-Deployments-Not-Leveraging-Secrets-Resources.html

The Kubernetes Software Development Lifecycle: Identifying & Mitigating Risks

Developing cloud-native applications with Kubernetes enables novel workflows that increase agility and innovation. The price for these novel workflows, however, is increased security risk. Below, we outline these risks and examine the Kubernetes development lifecycle to identify where vulnerabilities are introduced and how enterprises can mitigate them.

The lifecycle of a cloud-native application can be divided into four stages:

1. Development
2. Distribution
3. Deployment
4. Runtime

STAGE 1: Development of Cloud-Native Applications With Kubernetes

Traditionally, much of the security hardening for applications was done after the development stage. Once development was completed, security teams would assess the code, identify risks and vulnerabilities, and then send the code back for remediation. With cloud-native applications, teams are working in parallel to deliver new releases frequently, and there is simply no time to go back and forth between the security professionals and the software engineers.

Enterprises must seek efficient and effective ways to instill security into the development stage to ensure competitive software development speeds. This practice of shifting security responsibility "left" into the development stage means developers need to learn some new habits and skills to ensure security.

Kubernetes Secrets

Kubernetes pods sometimes contain sensitive information such as usernames, passwords, and SSH keys. Developers may hard code sensitive information and systems as a shortcut during development, or because they mistakenly trust that the containers within the workload are safe. When credentials are coded in the open like this, they are vulnerable to being seized and exploited by hackers. Kubernetes has a built-in solution to protect sensitive information called Secrets. Kubernetes Secrets allows engineers to encrypt passwords and other data so that it can only be accessed by authorized users and processes. However, **a study by Alcide in 2019 revealed that 89% of deployments were not using the Kubernetes Secrets**, leaving sensitive information exposed.

Development teams should train engineers to use Kubernetes Secrets as a matter of habit, but enterprises shouldn't stop there. **Gartner estimates**[2] that through 2022, 95% of security breaches will be the result of human error. Companies should provide developers with an automated monitoring tool to scan and detect sensitive information that has been left coded out in the open and flag it for immediate remediation.

Supply Infiltration

Enterprises that build Kubernetes-based cloud applications have other means to boost efficiency besides parallel development teams. One method is to utilize off-the-shelf and third-party code, primarily open source, to speed up the development process. A **study by Sonatype**[3] estimates that as much as 80% to 90% of the code in cloud-native applications contains open source components. Enterprises concerned with security must expand their coverage area to focus not only on the security practices of the programmers, but also to encompass the third-party code they integrate into their work.

The software supply chain is often taken for granted by security professionals, leaving it vulnerable to hackers seeking creative, indirect ways to compromise software. At its most benign, security vulnerabilities in off-the-shelf code might simply be inadvertent. At its worst, a trojan horse could bring malicious code into an enterprise's applications to steal data or hijack resources for activities such as distributed hacking or cryptomining.

At Rapid7, we advise our customers to begin with a clean slate and configure deployments in accordance with the best practices recommended by Kubernetes right from the start. The default configuration is typically optimized to make it easily accessible to developers, but a security-conscious team will begin by patching and updating all the software and then hardening the environment by locking down the configuration.

Rapid7's Key Recommendations:

Implement a starting environment that is fresh with clean code.

Install the latest patches and updates.

Configure according to Kubernetes' best practices for security.

We next recommend developers begin with a strategy that assumes all code might be compromised upon first contact. That way, if a container does become compromised, the rest of the application will be able to limit the spread of damage through tight security controls.

[2] Gartner: Is the Cloud Secure? https://www.gartner.com/smarterwithgartner/is-the-cloud-secure/

[3] Sonatype: 2017 State of the Software Supply Chain Report: https://www.sonatype.com/2017-state-of-the-software-supply-chain-report

Consider implementing the following practices:

1. Use admissions control in production to limit noncompliant resources from being admitted to the cluster.
2. Reduce the runtime privileges of workloads and avoid running them as root or with any elevated privileges.
3. Run workloads with an immutable file system.
4. Apply segmentation and isolation policies based on the workload at runtime.
5. Apply network policies.
6. Control network access to worker nodes.

Developer and Security Coordination

Organizations shifting their security responsibilities "left" into the developer's domain encounter a lot of human friction that slows down software development. Developers can't be expected to be security experts. Oftentimes, the dev team will face headwinds when trying to learn how to implement more secure code that will pass the security team's checks.

One effective way to introduce security expertise to the development group and increase compliance with security requirements is to designate a security liaison within the development team. The liaison can work with the security team to map out security requirements and then guide the development team in its implementation.

Both teams are efficiently served without the turbulence of retraining all the developers or blocking all code from passing through to production.

One effective way to introduce security expertise to the development group and increase compliance with security requirements is to designate a security liaison within the development team.

STAGE 3: Deploying Cloud-Native Applications With Kubernetes

The distribution stage is where the enterprise builds container images and artifacts. This is where developers should implement security infrastructure to ensure that the supply chain is secure and that the components used in the application code are not vulnerable. The distribution stage is where code scanning is implemented.

Container images in the software lifecycle pipeline require continuous, automated scanning and updates to ensure safety. Vulnerabilities, malware, insecure coding, and other risks must not only be prevented or detected, but they must be stopped from propagating throughout the application. Containers need to be checked, and then they must be cryptographically signed to ensure safety and prevent tampering.

At Rapid7, we advise our customers to secure these potential areas of security risk in the distribution stage of the Kubernetes software lifecycle.

Scanning & Hardening

The distribution stage is an excellent opportunity to ensure the security of container images and artifacts before putting them into production. By confirming security or identifying vulnerability at this stage, developers can remediate and harden images and artifacts so they can be deployed with confidence.

First, developers should employ automated scanning tools to establish that software has up-to-date patches to minimize risk in production. Automated scanning will also help detect malware and compromised open source code that may have infiltrated the code. At this stage, developers should also check for regulatory compliance to PCI, GDPR, and other relevant laws and standards.

In addition to detection of risks or malware, developers must install a robust procedure for remediating or hardening security flaws. Vulnerability scanners must provide results that are actionable for security teams.

Build Leaner Containers

As it turns out, vulnerable code in container images is not always exploitable. Containers may harbor security flaws or even malware in superfluous code that will never be executed in production. Such code is not dangerous to the application, but its presence will trigger time-wasting security alerts during scanning.

In the rush to complete container images and push them to production, developers may inadvertently include superfluous code. Rapid7 therefore recommends creating a well-defined process for building container images that will avoid unnecessary code that will generate false positives during security scans. By building leaner containers, developers will reduce the time they spend on tracking irrelevant threats.

STAGE 3: Deploying Cloud-Native Applications With Kubernetes

The deployment stage is when the enterprise conducts final checks before putting code into production. Essentially, the code is considered complete and secure, and the enterprise is executing a “pre-flight” checklist to ensure it complies with the enterprise’s security and regulatory compliance policies.

Security Drift Checks

As developers create code, they may bypass security protections or loosen access restrictions for testing purposes. Too often and too easily, these protections are not restored before the code is deployed. It is important to scan the code for security drifts that may have been introduced during development and prevent the drifts from going into production.

Employ a scanner that can detect if security configurations conform to the desired standard for all workloads, such as the use of Kubernetes Secrets resources, RBAC levels, and confirming that deployment packages are not configured to run with elevated privileges. The automated scanner must be able to detect security issues and automatically prevent them from going into production.

Compliance Checks

Enterprises face significant regulatory risks as countries enact protections for their citizens conducting business online. PCI, GDPR, CCPA, and other regulations provide guidelines to companies seeking to avoid fines and liability. **In a report by Verizon** on payment security, about 54% of companies in Europe and 61% of companies in America fail to comply with PCI DSS, the Payment Card Industry Data Security Standard.

Rapid7 has found that companies predominantly don't use Kubernetes Secrets resources. This shows why it's important to have automated systems in place to identify security and compliance risks. We therefore recommend enterprises employ an automated tool for scanning code during these pre-flight checks. Compliance for Kubernetes workloads takes on extra significance since standards such as PCI were not written with Kubernetes in mind. As a result, regulations often reference traditional server architectures, forcing developers to ensure Kubernetes compliance by determining the cloud-native analogs needed to comply with regulations.. Due to this, Rapid7 recommends compliance scanning tools that are purpose built for Kubernetes.

The runtime stage is where developer and security teams, having identified and remediated all possible known threats and vulnerabilities, encounter the unknown: the undetected or novel vulnerabilities and intrusion strategies of hackers.

Examples of security and privacy compliance checks:

- Check for firewalls and firewall configurations to protect cardholder data.
- Prohibit direct public access from the internet to any system component in the cardholder data environment.

STAGE 4: Runtime for Cloud-Native Applications With Kubernetes

The cloud-native runtime environment encompasses the following:

- The infrastructure where workloads are executed.
- The service layer abstractions that help containers and microservices interact.
- Security controls and network traffic monitoring from the runtime providers.

The runtime stage is where developer and security teams, having identified and remediated all possible known threats and vulnerabilities, encounter the unknown: the undetected or novel vulnerabilities and intrusion strategies of hackers.

[4] Verizon 2020 Payment Security Report: https://www.verizon.com/business/resources/reports/payment-security-report/

At this point, enterprises must utilize advanced and automated monitoring solutions to detect unauthorized activity and block intruders. Here are some areas to watch.

Audit Logs

The elastic and ephemeral nature of Kubernetes makes it hard to trace the paths and actions taken by hackers. Containers that were used in an attack might have only existed for moments during runtime. Kubernetes Audit Logs provide a detailed record of what's happening in a workload, and are invaluable for reconstructing past activity and observing workloads in real-time for abnormal or suspicious activity. Be aware that audit logs are not always enabled in some managed Kubernetes deployments, so organizations should be sure to check the audit log status.

Audit logs are challenging to understand and time consuming to analyze manually. Therefore, Rapid7 recommends automation and machine learning when monitoring or analyzing logs. Automated log monitoring tools can watch Kubernetes Audit Logs for risky behaviors. For example, an enterprise might set up an exception where nobody is allowed to dump compliant data from the log. Compliance officers would then be notified if this occurs, as it could be a data breach.

> **"...machine learning can develop a picture of what typical, secure behavior looks like in the workload and use it to identify outlier behaviors that may indicate nefarious activity."**

More broadly, machine learning can develop a picture of what typical, secure behavior looks like in the workload and use it to identify outlier behaviors that may indicate nefarious activity. For example, in the case of credential theft, a Kubernetes Audit Log monitor would be able to identify if an attacker logs into the Kubernetes cluster in a fashion that doesn't match typical activity for the team member.

Kubernetes Audit Log monitoring detects:

- Attempts to gain access to endpoints
- Unusual cluster behavior
- Stolen credentials
- Misconfigured RBAC
- Exploitation of vulnerabilities in Kubernetes API servers
- Violations of security policies

Microservices

Since Kubernetes decouples containerized applications from the underlying infrastructure it runs on, workloads are much more flexible as to where in the cloud they may be executed. This provides enterprises with profound scalability and agility. But, due to the distributed nature of Kubernetes-based applications, Kubernetes does not easily fit into the definitions of security perimeters used by traditional security software, such as firewalls. With Kubernetes, there is not a clear definition of what falls within or outside of the "perimeter," which makes securing these applications tricky.

At Rapid7, we solved this challenge with microservices firewalls. We enable developers to package security policies into agents that can be bundled with microservices during the build process. During runtime, the agent goes wherever microservices are running, and provides visibility across cloud infrastructure, enabling administrators to see how applications are deployed and to manage security for far-flung containers. Microservices provide awareness into where applications are running and how they should act security wise, which they share with administrators. With a microservices firewall, administrators can search for, isolate, control, and enforce specific security policies in real time.

The microservices firewall also extends protection to the APIs used by microservices to communicate with each other. Machine learning algorithms watch network traffic through APIs in real time to detect suspicious behavior and prevent suspected breaches from propagating. Finally, the rich data provided by the microservices firewall can then be shared with developers to create a feedback loop to improve security for future versions of the microservices.

Cloud-Native Security Must Match the Kubernetes Software Lifecycle

Kubernetes and the software development lifecycle it enables brings speed, agility, and elasticity to cloud-native applications. Kubernetes's benefits also come with unique responsibilities for securing workloads, containers, infrastructure, as well as the organization itself. In order to realize the benefits of a Kubernetes-based infrastructure, enterprises must also evolve their security practices.

Enterprises on a Kubernetes journey cannot apply the traditional security solutions that once protected on-site servers or VMs, as those are not designed for the unique development practices and distributed environments with which Kubernetes excels. Enterprises must expand their existing security programs to cover the entire Kubernetes software lifecycle with solutions that are intentionally built to solve for the increased scale and complexity of cloud-native environments.

Rapid7 has the experience and technology to protect enterprises using Kubernetes with InsightCloudSec. The Kubernetes security functionality built into InsightCloudSec is designed to address the specific needs of DevOps teams working with Kubernetes. Our years of experience have helped us gain a deep understanding into how Kubernetes works, how developers work with Kubernetes, and how attackers attempt to penetrate Kubernetes-native applications. We invite you to join us to learn how InsightCloudSec integrates Kubernetes security into a full cloud-native security solution.

Are you on a Kubernetes journey? **Contact us today** to see how InsightCloudSec can help you drive innovation through continuous cloud security and compliance.