The Essential Guide to

Cloud-Native Security

CAPSULE8

Contents

Chapter 1

What is Cloud-Native Security?

Chapter 1

What Is Cloud-Native Security?

Cloud-native security protects applications designed and built on the cloud. On a higher level, “cloud native” is a fundamentally new approach to application design and deployment, leveraging native cloud capabilities like auto-scaling, continuous deployment, and auto-management. It is an open source approach leveraging IaaS capabilities (e.g. AWS, Microsoft Azure, and Google Cloud) to create new tools and services that are more responsive in the age of the customer. From a developer’s perspective, cloud native means shipping fast and often—without sacrificing reliability. From an operations perspective, cloud native means automatic management and massive economic gain in resource consumption.

Cloud-native applications are typically built using a microservices or container-based approach running on Linux. These applications are designed to be lightweight, flexible, and focused on single tasks. Ultimately, they’re like smaller building blocks that are pieced together to achieve speed, scalability, and efficiency savings that you simply can’t get with a traditional monolithic architecture.

Simply put, cloud native *IS*:

Microservice-centric:

A cloud-native application today must be built on microservices. Traditional monolithic applications do not support continuous deployment, continuous update, and auto-scaling, which are some of the core benefits of cloud.

Portable:

Being cloud-native means that your applications should not be tied to a specific cloud platform. In a cloud-native environment, interactions between the application components are done via standard service APIs. Operations tasks such as deployment, monitoring, and workload management are all conducted via either open source or common functions that can run across different clouds.

Automatically managed:

Common workload management tasks such as deployment, updates, monitoring, and scaling are all done automatically. Manual tasks, including manual security analysis or configuration, are the exception rather than the rule.

And cloud native is *NOT*:

Replicating your on-premises setup in an IaaS cloud:

Don't think just because you migrated parts of your application environment to run in some EC2 instances, you are cloud native.

Packaging your monolithic applications with APIs:

Adding APIs to your monolithic applications do not make them cloud native. You must re-architect your applications to be distributed, service driven, and on-demand. This also applies to data and workload management services.

Chapter 2

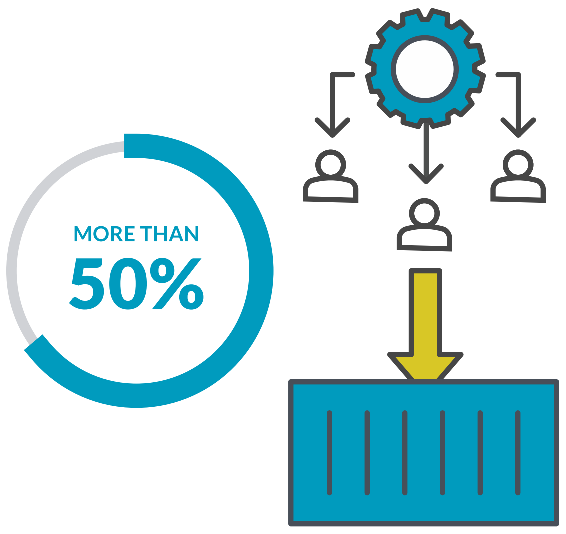
Understanding the Cloud-Native Landscape

Chapter 2

Understanding the Cloud-Native Landscape

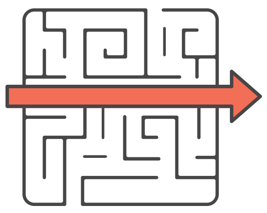
What does the cloud-native landscape look like today?

According to Gartner, “By 2018, more than 50% of new workloads will be deployed into containers in at least one stage of the application lifecycle.”

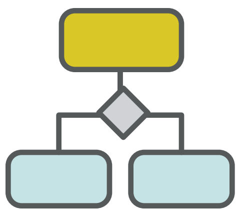


More than 90 percent of the Fortune 500 companies today are already running Linux. While the cloud-native approach is a very recent phenomenon, change is happening rapidly—and it’s disrupting industries.

The current cloud-native landscape is dominated by three major trends:



1 Simplifying and automating existing IaaS capabilities: This means integrating workloads on AWS, Microsoft Azure, and Google Cloud platforms *to only run when required.*



2 Moving to a microservices architecture: One of the ways to understand microservices is to look at companies like Google, Netflix, eBay, and Twitter, who are early adopters of this model. Microservices deconstruct a single

application and break it down into smaller functional components. This greatly reduces overhead, while fundamentally changing the way applications are developed, deployed, and managed.

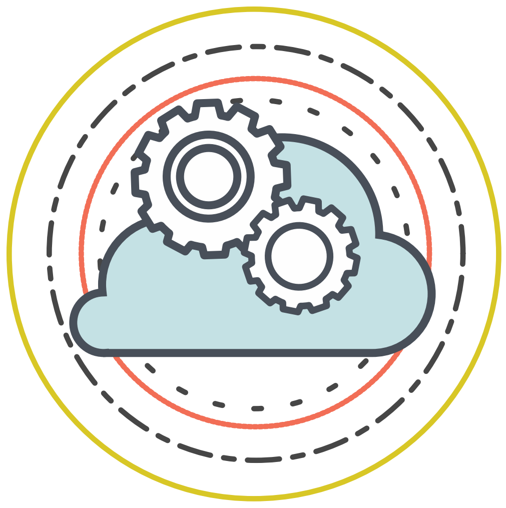
3 Developing event-driven computing: Event-driven computing, also known as serverless, allows you to run code without provisioning servers first. It provides easy management of computing resources and a much-needed productivity boost by decoupling software development from server administration. Services like AWS Lambda, Azure Functions, and Google Cloud Functions enable this transition.

Chapter 3

The Benefits of a Cloud-Native Architecture

Chapter 3

The Benefits of a Cloud-Native Architecture



At this point, the business benefits of the cloud-native approach are fairly clear. A cloud-native architecture enables speed, business efficiency, nearly limitless computing power, and a level of scalability that can't otherwise be achieved with the traditional on-premises IT model, or by simply migrating the "old way of doing IT" to a public cloud vendor like AWS. It not only requires a fundamental shift in the way applications are designed and deployed but also engenders far-reaching impact on team and organization culture and fundamentally changes the way your business responds to the shifting demands of the marketplace.

Organizations that embraced a cloud-native architecture saw increased elasticity and a better ability to support continuous deployment. Elasticity and continuous deployment are some of the most critical functionalities in the cloud native era because they enable limitless scalability and business agility.

Chapter 4

Cloud-Native Security

Chapter 4

Cloud-Native Security

But there's a catch—cloud-native security.

Cloud native is a fundamentally new and exciting approach to designing and building applications. This, however, also raises a completely new set of security challenges. For example, when you move to a microservice model, end-to-end visibility, monitoring, and detection become more complex and difficult to execute. Similarly, there is currently no solution focused on protecting the entire Linux stack, even though most microservices applications run on Linux. In addition, conducting security operations and making security decisions at the "service" level is an unfamiliar territory for the security industry.

This means that traditional security assumptions, such as the presence of an agent, a network perimeter, and end-to-end visibility, may not be valid anymore. Many of the security capabilities that we learned to depend upon, such as server instrumentation, may also be ill-suited for the cloud-native environment.

Being cloud-native leads to a radically different approach to application development and deployment, and to infrastructure management. The same is true for cloud-native security. A reimagining of security must take place for cloud-native applications, or we risk cannibalizing the benefits of cloud computing.

Chapter 5

Are Cloud Security Risks Different?

Chapter 5

Are Cloud Security Risks Different?

While enterprises begin experimenting with the benefits of cloud-native applications, the practical side of cloud-native security and management is less understood. Are the implications of security truly different in a cloud-native environment than a more traditional one? How do cloud security risks impact your security strategies and controls?

Some of the top cloud-native security considerations are:

1. Continuous delivery requires continuous security: As microservices and containers replace monolithic and traditional multi-tiered applications in cloud-native environments, software delivery and deployments are becoming continuous. Organizations like Amazon and Target are doing hundreds of deployments per day. In these types of environments, security checks must be lightweight, continuous, and embedded into the deployment tool chains, or they risk being bypassed.

2. Protecting server workloads is imperative: Traditional enterprise security is about securing the endpoints, segmenting the network, and protecting the perimeter. In a cloud-native environment, you may not be able to rely on fixed routes, gateways, network perimeters, or even the presence of an agent—the forefront of threat is now at your data center. Your server workloads are more exposed to the surface of attack than before. Shifting focus to securing the data center and server workloads is therefore imperative.

3. Run-time detection at speed and scale: In a microservices model, end-to-end visibility, monitoring and detection become more complex and difficult to execute, especially when deployment and upgrades are continuous. Attack detection needs to work dynamically (e.g., less reliance on static signatures), scale in real-time, and do so without jeopardizing the performance and stability of the production environment.

4. Hybrid stack protection: Some microservices applications run in containers on a virtual machine, while others are on bare metal Linux. But, today's security functions protecting the host, the VM layer, the container, and the applications are often separate and bereft of integration. This approach introduces complexity and unnecessary barriers to executing real-time security response and actions. For instance, would you deploy a mission-critical container to a VM that needs patching? But how would you know that VM needs patching if you don't also have the VM-level visibility? There is a distinct need to integrate visibility, monitoring, and protection across this hybrid stack including the Linux host, VM, containers, and finally the application and its services.

Chapter 6

What Does the Ideal Cloud-Native Security Platform Look Like?

Chapter 6

What Does the Ideal Cloud-Native Security Platform Look Like?

Before enterprises design a cloud-native security strategy, they need to consider these additional requirements:

A high degree of security automation: Traditional alert-based security operations simply cannot keep up with the near-limitless scale, and the far more dynamic nature, of cloud-native systems. As such, manual workflows are simply not an option. Automated detection and response at scale is a must-have requirement for cloud-native security.

Design with “chaos” in mind: In a microservice architecture, any function may involve many software components stitched together at runtime. From a security standpoint, this means detection logic and controls cannot rely on a prior understanding of the operational state and security health. Rather, cloud-native security must embrace chaos engineering principles—experiment proactively, test often, and remediate fast.

Detect fast, contain locally, and recover rapidly: Cloud-native applications are ultimately distributed computing applications. In such an environment, it is often not possible to execute global security decisions in a timely manner. Thus, you would do well to prioritize actions that allow you to detect fast, recover rapidly, and contain impact locally before system-wide, malicious behaviors manifest. Even if your security decision is not 100% precise (they rarely are), acting locally and recovering fast will give you a more resilient system overall.

Finally, what features should your cloud-native security solution have? We believe the top runtime feature sets are as follows:

Visibility and decision support across the hybrid stack: In the cloud-native data center, you need visibility across the host, VM, containers, applications, and API services, even when the applications are distributed, services dynamic, and containers short-lived. Information gathered on these different layers should go into an engine which enables a real-time decision-making process.

Rapid detection and response functions that minimize the “blast radius”: Your cloud-native security solution must ensure that if there was a security incident, the fallout is minimized and contained. This logic leads to rapid decision-making and smart controls that stop the malicious behavior before it creates irrevocable damage. In a cloud-native world, it is entirely possible that a smart detection system can spot the onset of an attack and affect local controls.

Continuous monitoring and support for investigation at scale: Security investigation for cloud-native workloads can be complex because of all the distributed components and API services, so monitoring and security investigation must minimize the impact to performance and demand for storage. This necessitates a monitoring/investigation architecture that is distributed, lacks system bottlenecks, and can scale with the workloads.

Detect fast, contain locally, and recover rapidly: Cloud-native applications are ultimately distributed computing applications. In such an environment, it is often not possible to execute global security decisions in a timely manner. Thus, you would do well to prioritize actions that allow you to detect fast, recover rapidly, and contain impact locally before system-wide, malicious behaviors manifest. Even if your security decision is not 100% precise (they rarely are), acting locally and recovering fast will give you a more resilient system overall.

As containers and event-driven computing replace physical servers and first-generation virtual machines, security must find the right insertion point in this new setting to gain visibility and mitigate risks, while enabling innovations and adjusting to the complexities of continuous delivery in a cloud-native environment.

Chapter 7

The Evolution of Attack Detection

Chapter 7

The Evolution of Attack Detection

Traditional detection is a human exhaustion exercise.

Attack detection in a traditional security operations center (SOC) is based on investigation and analyzing alerts and event logs. When you have many security devices generating logs and alerts, a typical SOC may see thousands of alerts and Terabytes of logs per day.

Because of this, a SOC analyst may have to handle hundreds or thousands of alerts a day to keep up with the volume. Analysts are overwhelmed, and yet SOCs still lag in terms of alert handling and attack detection.

In addition, many SOCs do not have effective metrics to measure true efficacy or success of operations. When a SOC measures alert handling rate per analyst, it is essentially running on human fumes, and the only way to scale up is by adding more humans to fuel that fire. We all know where that strategy goes.

Modern detection is as much about engineering as it is about security analysis.

This is a very different treatment to the attack detection problem—rather than solving the problem with analysts and more analysts, you use engineering principles to develop logics and tools that not only automate detection, but also continuously update and deploy new detection and response logics.

For instance, one of the logics for an attack detection program is the automatic contextualization of alerts—pulling relevant information from endpoints, network logs, Active Directory, threat intelligence feeds, etc. Executing this logic will reduce the amount of time a human analyst must expend to query different devices and interface with different tools just to gain context for the alerts.

Another example might be a rule to automatically trigger a second-factor authentication workflow or force a password reset if a risky event is detected.

Chapter 8

The Evolution of Software Engineering for Attack Detection

So which software engineering principles make sense for attack detection? Below is a list that maps well-known engineering practices with modern detection engineering.

Unit test the creation of new detection rules:
Any time a new detection rule is created, you must practice extensive unit tests. You may write scripts or simulations to directly attack the detection rules or its assumptions.

Integration test detection workflows:

Detection workflows are often complex and may impact multiple teams and infrastructure. You should employ integration testing principles to ensure the quality and reliability of attack detection workflows. More teams should be testing detection tasks the same way you'd test a build pipeline or an automated configuration management workflow.

Tool the most frequent scenarios:
Just as software developers encapsulate well understood tasks into "subroutines," the frequently executed detection tasks/workflows can be codified into automated tools or programs to eliminate manual tasks and improve efficiency.

Ensure a continuous feedback loop and continuous improvement:
Your detection infrastructure needs continuous improvement. The first step to achieving that is establishing feedback loops with infrastructure components, security devices, and even with different teams and users. These feedback loops will allow you to create an assurance process to improve the quality and precision of your attack detection programs.

Manage repository and accountability:
Not unlike any well managed software engineering organization, you should house detection rules and codified workflows in a centrally managed repository. Efficacy metrics for rules and detectors should be tracked to allow continuous improvement.

Chapter 9

Detection Engineering in Cloud-Native Security

Chapter 9

Detection Engineering in Cloud-Native Security

To perform detection engineering, cloud-native security tools must, at the very least, provide these capabilities:

Ability to handle cloud native infrastructure:

Detection technology must be tooled to handle cloud-native components like containers, serverless, and microservices. At the same time, it must work seamlessly with detection technology designed for virtual machines, physical servers, and traditional networks.

Effectively reduce and normalize security alerts:

Because cloud-native workloads can be ephemeral, alert volumes may be higher than that of a traditional system and can easily overwhelm even the most sophisticated modern detection infrastructure. Thus, the task of reducing and normalizing alerts into meaningful signals is essential in a cloud-native security system.

A critical scaling factor:

Moving from manual handling of alerts to detectors and logics renders a “scaling factor”—a bulk of the alerts covered by the detectors are either discarded or acted upon without being specifically reviewed. There is no other place the impact of this scaling factor is more pronounced than in a cloud-native environment because of the sheer scale and velocity of such systems.

Modern detection engineering requires the adoption of engineering principles to security analysis. In cloud-native security, this practice becomes existentially critical—without it, attack detection will be untenable.

Looking for a comprehensive cloud-native security solution? Capsule8 has got you covered across your entire production environment—containers, virtual machines, and bare metal.

REQUEST A DEMO