Hillstone NETWORKS

WHITE PAPER

Hillstone Networks
CloudArmour Solution

How to Secure Your Cloud Workloads

Moving from VMs to Container Orchestration

Modern enterprise network environments are increasingly transforming to be cloud-based, where both applications and data storage are hosted in a cloud—and often multi-cloud—environment. The attack surfaces and security protection requirements of software in distributed cloud environments are vastly different from traditional network architectures where applications and data were hosted on enterprise-owned servers in on-premises data centers.

Along with the business environment transformation, there is a parallel trend in the modernization of software development processes and environments. Applications have evolved from single-server software installs, to virtual machine (VM) server-independent environments, and more recently to container-based technology. In these modern cloud-native, container-based architectures, the security mechanisms and protections for your business applications must evolve in line with software development methods and tools, as well as with the threats that exist during deployment, orchestration and day-to-day operation.

Hillstone's CloudArmour is a cloud workload protection platform (CWPP) that provides comprehensive protection for all cloud workloads, including containers, VMs and other execution environments. CloudArmour provides enterprise IT teams with cloud-native container security capabilities for the Kubernetes environment.

Containers

IT development teams increasingly implement applications using container technology rather than VMs. Container design deconstructs the implementation of business-critical applications into small, independently maintainable micro-units. Each micro-unit can be updated, iterated, version-tracked and debugged separate from other micro-units.

Container implementations scale easily, and consume only the resources and performance needed to meet current demand. Container architectures provide important operational benefits—they are much more flexible, portable, scalable, maintainable and efficient than traditional implementations.

Contents

Introduction: Moving from VMs to Container Orchestration 2
- 2 Containers
- 3 Microservices
- 3 Container Orchestration

Why Do We Need Cloud Workload Protection? 3
- 4 The Attack Surface of Container Applications
- 6 Container Security Requirements

Solving Container Security Challenges 6

Hillstone's CloudArmour CWPP Solution 8

Hillstone's CloudArmour CWPP Core Features 8
- 9 Container Asset Management
- 10 Container Asset Traffic Visualization
- 10 Network Micro-Segmentation of Cloud Workload
- 11 Process Behavior Monitoring of Cloud Workload
- 11 Vulnerability Scanning for Cloud Workload Environment
- 12 Compliance Checks for Containers, Container Images, K8s, and Nodes

The Value of Hillstone's CWPP Solution 12

Microservices

A micro-unit design lends itself to a microservices applications architecture. Unlike the tighter coupling of traditional monolithic application architectures, a microservices architecture is loosely coupled and significantly more flexible. Docker is arguably the most widely used container platform—Docker technology wraps a layer around the container runtime standard Containerd.

Developers build business applications and services in containers, often by creating multiple independent but interactive microservices. Microservices can exist in a container environment as a single instance, or share the same network stack in a group known as container pod. Both single-instance containers and container pods communicate with each other at the network layer.

Container Orchestration

Running containerized workloads or services require significant operational effort, which are typically automated by an orchestration service. The orchestration of container creation customarily involves networking together the container instance or pod, assigning IP addresses, and providing domain name resolution capabilities for the microservices within the container environment. Container orchestration is essential because the number of container applications to be maintained is routinely exponentially larger than in a VM environment. This scale demands automatic—not manual—orchestration to effectively manage container services.

The Kubernetes platform (also known as K8s) has become the standard for container orchestration and is typically used by IT teams in production environments. K8s includes:

• A complete container asset orchestration program for setting up networking

• Domain name resolution for container pods and services

• Life-cycle, capacity expansion, availability and performance management for container assets

Why Do We Need Cloud Workload Protection?

Enterprises increasingly use container technology to build business-critical services, while hackers continuously probe for vulnerabilities in containers and container orchestration platforms. Attacks on container services increase every year. Three typical attacks have already occurred by early 2022:

1) A new Muhstik botnet variant exploited unauthorized access vulnerabilities of the Kubernetes Kubelet API and Docker Remote API and successfully attacked cloud servers. Once infiltrated, Muhstik installs an IRC backdoor to establish remote control of the compromised system. Hackers can then issue and execute arbitrary commands, such as downloading and installing mining modules, as well as performing DDoS attacks.

2) The mining Trojan LoggerMiner attacked a cloud host by using SSH account information on the current host to launch attacks against other hosts. LoggerMiner also attempts to infect Docker containers on the current host.

3) The TeamTNT mining Trojan exploited an unauthorized access vulnerability in the Docker Remote API to attack cloud servers. After infiltration, TeamTNT hides processes, launches persistent attacks by installing scheduled tasks, and moves laterally using multiplexed SSH connections to infect other servers. TeamTNT also ties up a large amount of CPU resources and may crash the service system.

Legacy security mechanisms fall short in providing sufficient protection for the cloud-based open-source development and operating environments common in today's tool and application architectures. Figure 1 details how and where traditional security solutions lack the capabilities to protect modern dynamic software architectures.

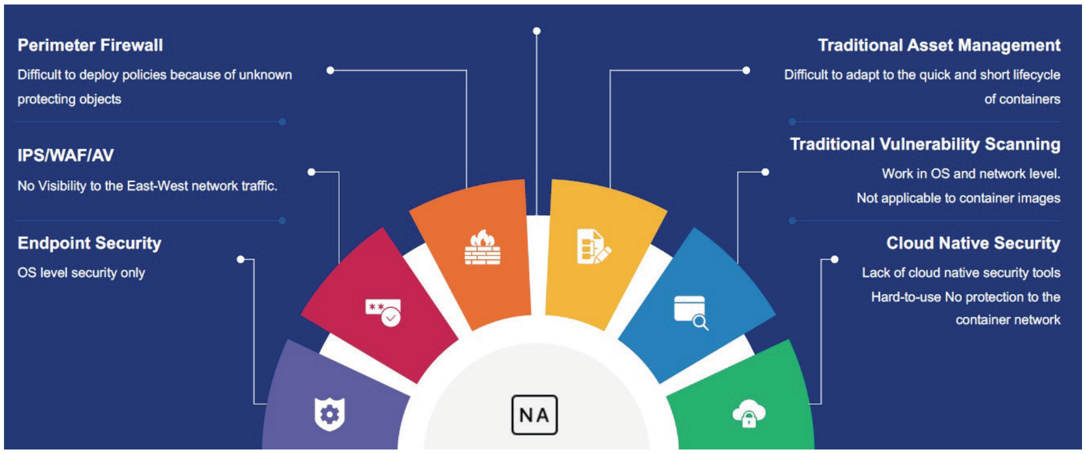
Perimeter Firewall
Difficult to deploy policies because of unknown protecting objects
IPS/WAF/AV
No Visibility to the East-West network traffic.
Endpoint Security
OS level security only
Traditional Asset Management
Difficult to adapt to the quick and short lifecycle of containers
Traditional Vulnerability Scanning
Work in OS and network level.
Not applicable to container images
Cloud Native Security
Lack of cloud native security tools
Hard-to-use No protection to the container network
NA

Figure 1: Shortfalls in Traditional Security Solutions

The Attack Surface of Container Applications

Security is paramount when using a container design to implement business-critical applications. Adoption of container technology in general, and Kubernetes in particular, continues to ramp up in the industry even while the understanding of vulnerabilities and attack surfaces present in container development and operation environments are not widely appreciated. Overall security challenges include both the host environment—as shown in Figure 2—as well as the container environment—as shown in Figure 3.

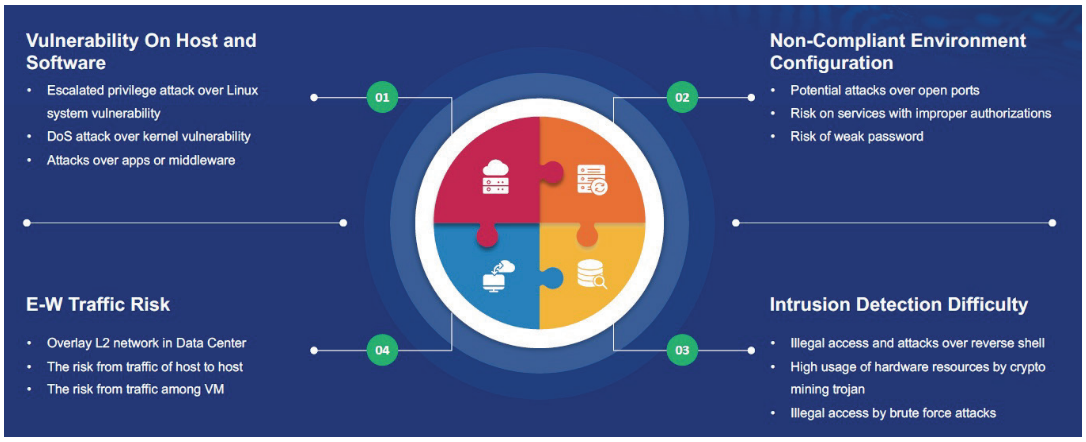
Vulnerability On Host and Software
Escalated privilege attack over Linux system vulnerability
DoS attack over kernel vulnerability
Attacks over apps or middleware
01
Non-Compliant Environment Configuration
Potential attacks over open ports
Risk on services with improper authorizations
Risk of weak password
02
E-W Traffic Risk
Overlay L2 network in Data Center
The risk from traffic of host to host
The risk from traffic among VM
04
Intrusion Detection Difficulty
Illegal access and attacks over reverse shell
High usage of hardware resources by crypto mining trojan
Illegal access by brute force attacks
03

Figure 2: Host Security Challenges

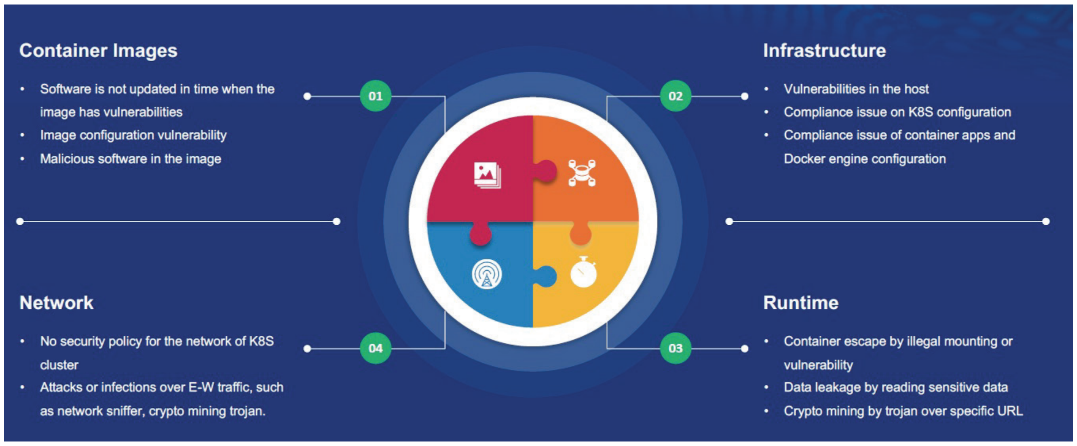
Container Images
Software is not updated in time when the image has vulnerabilities
Image configuration vulnerability
Malicious software in the image
01
Infrastructure
Vulnerabilities in the host
Compliance issue on K8S configuration
Compliance issue of container apps and Docker engine configuration
02
Network
No security policy for the network of K8S cluster
Attacks or infections over E-W traffic, such as network sniffer, crypto mining trojan.
04
Runtime
Container escape by illegal mounting or vulnerability
Data leakage by reading sensitive data
Crypto mining by trojan over specific URL
03

Figure 3: Container Security Challenges

In general, cloud workload protection is much larger in scope than traditional network security. The key challenges for creating a secure cloud native platform include:

• **Managing for Security:** Security in the three stages of container development, deployment and operation must be initiated up front and managed throughout the pipeline. Access privileges, credentials, repositories, image scanning and adhering to CIS (Center for Internet Security) guidelines are key in this effort.

• **Monitoring:** Code development and file programming paradigms are vastly different today from older development systems that involved only local workstations, hosts and repositories. Basic host monitoring is no longer sufficient. Cloud-based storage and various open-source-based tools are widely used and far more granular monitoring is essential to ensure there is no tampering or contamination of images or data. A container often shares the system kernel with its host machine. If the container's permission privileges are set too permissive, it can allow malicious code to penetrate and obtain control of the host machine.

• **Data Persistence:** To facilitate loosely coupled operation between containers (without affecting each other), various services may use their own private database resources, increasing the overall attack surface.

• **Networking:** The container east-west network layer is generally invisible and uncontrollable by traditional network security tools, and interservice interactions between containers is spread across many different IP addresses. The communication between containers represents a significant attack surface, and the networking environment must be tightly integrated for Kubernetes.

• **Container Life Cycle Management:** Outdated supply chain code or libraries from vendors may be contaminated, risking backdoor exploitation. Rapid deployment focuses on the latest image, while older versions are disregarded but seldom deleted. As the development environment iterates rapidly, older versions of code or tools still exist in the repositories and may create risks.

• **Container Orchestration:** Any insecure default configurations or excessive developer privileges can introduce vulnerabilities in the orchestration platform, typically Kubernetes, that can be leveraged as attack vectors.

Container Security Requirements

The attack surfaces can be grouped into three stages, each of which requires different layers of security capabilities to protect and defend against threats.

- Development (coding, and continuous integration, delivery, and deployment CI/CD)
- Deployment (static security)
- Operation (dynamic security)

Figure 4 shows the breakdown of container security requirements spanning all three stages of the container lifecycle. Key among these are the need for image scanning, repository security, safe configurations, managing access control and developer credentials, securing host and kernel processing, and protecting container inter-process communication.

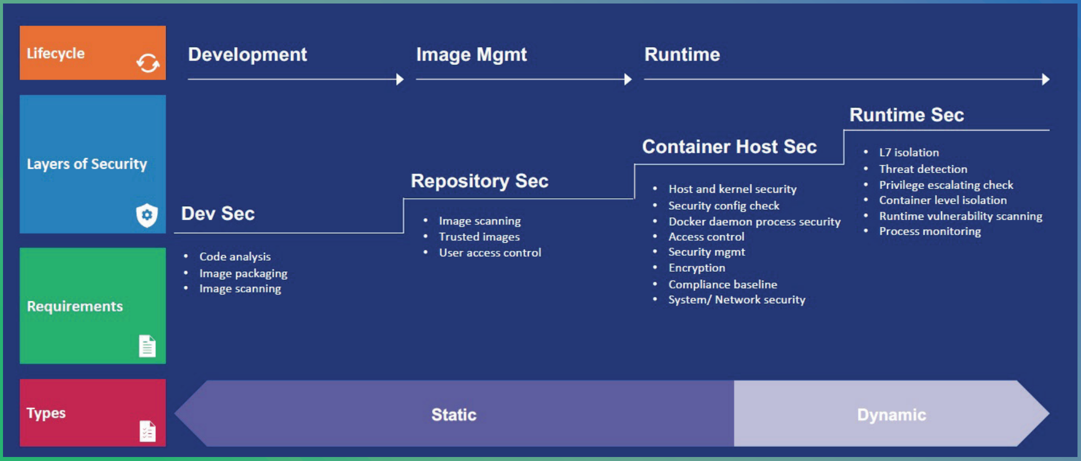

Figure 4: Container Security Requirements

Solving Container Security Challenges

Container security technology is often considered part of host security, yet traditional host security products are unsuitable for container scenarios.

1) Terminal agent software. The terminal agent software of traditional host security products is manually installed and deployed. A container environment typically has a large number of nodes, making manual installation per node impractical in terms of staff workload. The terminal agent software cannot automatically expand along with the container node—manual adjustments are required.

2) Orchestration management platform. Traditional host security products are usually unable to connect to the container orchestration management platform. Without such a connection, monitored host security events can be neither matched to container assets, nor can security policy management be implemented around container assets.

3) Scope. Critical parts of container security—such as the need for container image vulnerability scanning—are beyond the purview of traditional host security.

An effective container security solution—as shown in Figure 5—must reflect the security requirements identified in the previous section. The solution must be designed to cover all three of the stages: development (code and CI/CD); deployment (static security); and operation (dynamic security). Additionally, it must provide capabilities including code security, image security, container engine and orchestration management platform security, container runtime security, network security, and application security.

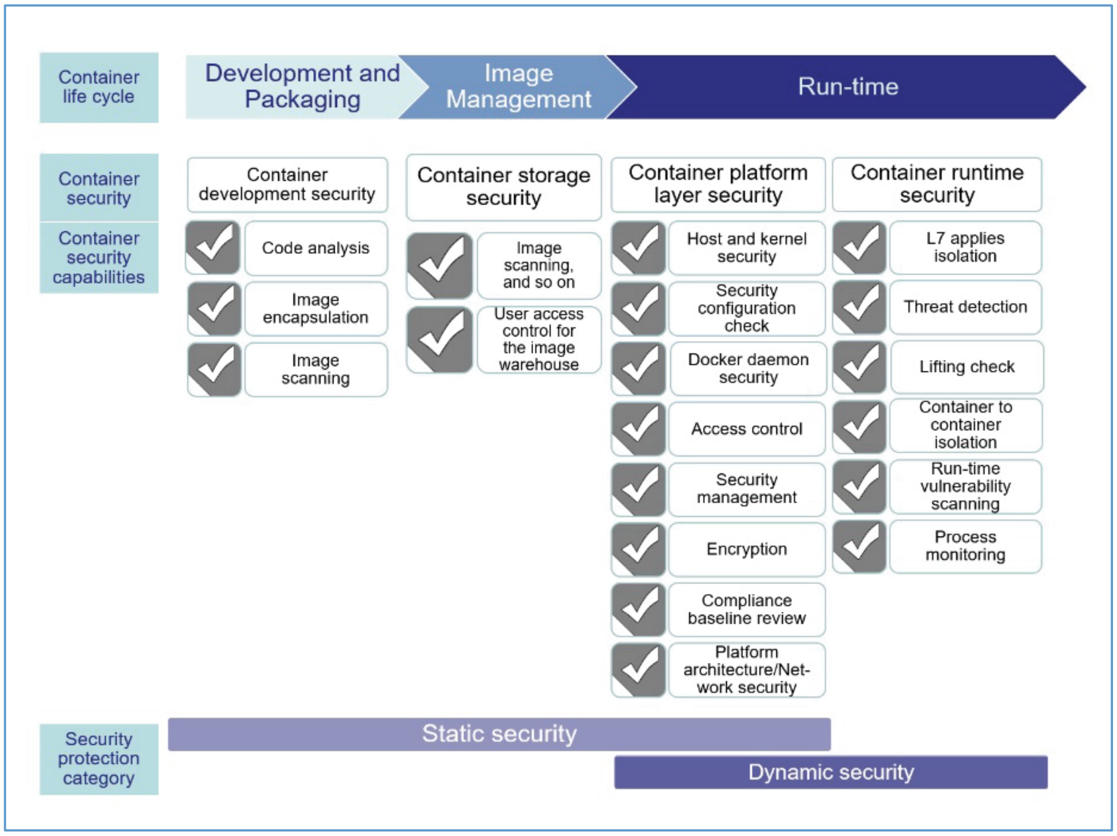
Container life cycle
Development and Packaging
Image Management
Run-time
Container security
Container security capabilities
Container development security
Code analysis
Image encapsulation
Image scanning
Container storage security
Image scanning, and so on
User access control for the image warehouse
Container platform layer security
Host and kernel security
Security configuration check
Docker daemon security
Access control
Security management
Encryption
Compliance baseline review
Platform architecture/Network security
Container runtime security
L7 applies isolation
Threat detection
Lifting check
Container to container isolation
Run-time vulnerability scanning
Process monitoring
Security protection category
Static security
Dynamic security

Figure 5: Anatomy of a Container Security Solution

Typical container security solution architectures include a distributed security container and centralized management. The distributed security container is deployed as a DaemonSet, allowing Kubernetes to expand and shrink the security container as well as monitor status and failure recovery. Container security products are focused on protecting cloud-native business architectures, therefore a cloud-native foundation for the security product itself is advantageous.

Hillstone's CloudArmour CWPP Solution

CloudArmour is a cloud workload protection platform (CWPP) that provides comprehensive protection for all cloud workloads, including containers, VMs and other execution environments.

Figure 6 details the architecture of Hillstone's container security product, CloudArmour—a solution that embraces a distributed security container as well as centralized management. CloudArmour uses Kubernetes cloud-native technology as its direct foundation, and consists of three principal aspects:

1) **The security guard consists of distributed secure containers** deployed as DaemonSets under Kubernetes, allowing scaling, condition monitoring, and failure recovery of secure containers.

2) The **centralized management** component leverages the resilient and reliable service-clustering capabilities of Kubernetes.

3) The **policy configuration and data reporting** between the management component and the security container uses Kubernetes' ETCD based messaging mechanism. This architecture allows the effective management (by the central management component) of an exponentially larger number of security containers than the number of devices that can be managed by traditional network management components.

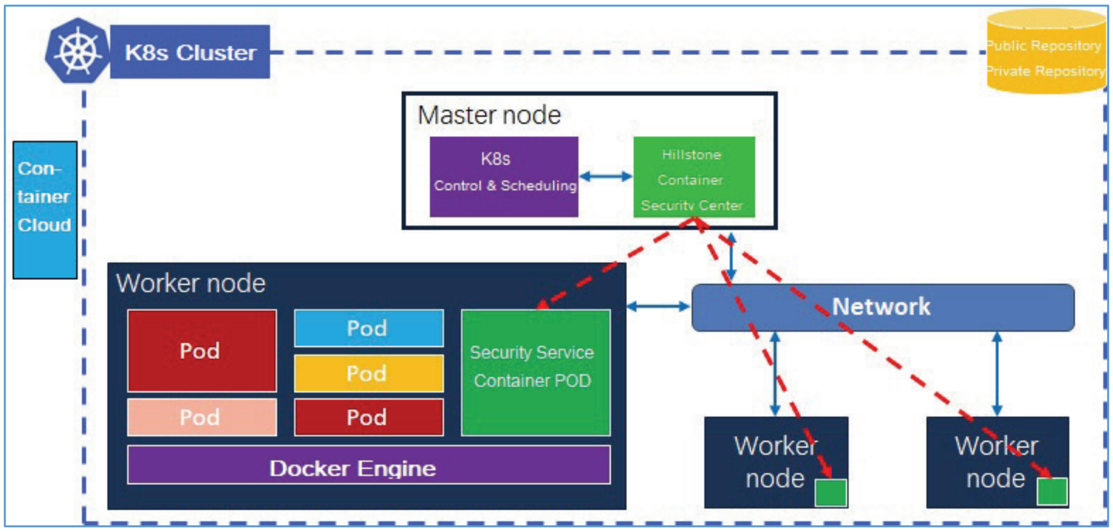

Figure 6: The Architecture of Hillstone CloudArmour

The security guard is the core component of the Hillstone CloudArmour solution. While CloudArmour supports both static and dynamic security, its design requires the least privileges possible. CloudArmour runs strictly as a user-mode process—it does not require either kernel or privileged mode—and leverages the host operating system's existing NET_ADMIN, SYS_ADMIN, SYS_PTRACE, and AUDIT_CONTROL kernel functions.

Hillstone's CloudArmour CWPP Core Features

Figure 7 provides a summary of CloudArmour's core features, each of which is further discussed in the sections following the figure.

- Container environment asset management
- Container asset traffic visualization
- Network micro-segmentation of containers and nodes
- Process behavior monitoring of containers and nodes
- Vulnerability scanning of container images and nodes
- Compliance checks for containers, container images, K8s and nodes

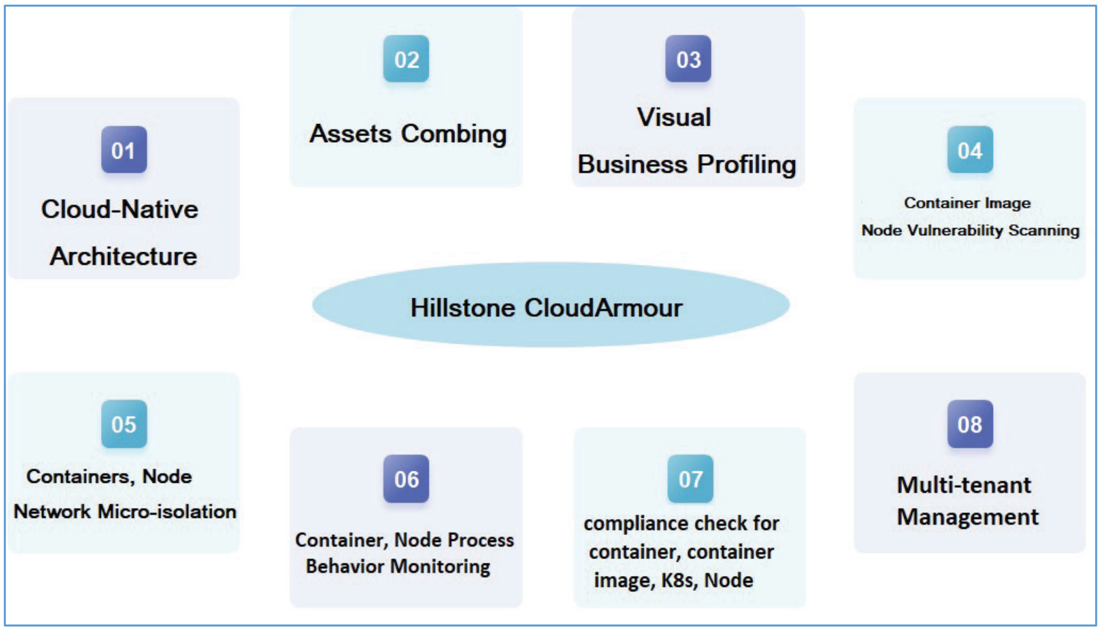
01
Cloud-Native Architecture
02
Assets Combing
03
Visual Business Profiling
04
Container Image Node Vulnerability Scanning
Hillstone CloudArmour
05
Containers, Node Network Micro-isolation
06
Container, Node Process Behavior Monitoring
07
compliance check for container, container image, K8s, Node
08
Multi-tenant Management

Figure 7: Core Features of Hillstone's CloudArmour Solution

Container Asset Management

In a container environment, asset objects are not tied to IP addresses; instead, the IP addresses of workloads often change dynamically. Protected objects and container security solutions like CloudArmour therefore must follow the asset organization logic imposed by the business environment.

Kubernetes orchestrates container pods and organizes different types of assets, primarily Pods, Deployment, DaemonSet, Services, Ingress, Namespace, Cluster, and others. CloudArmour's centralized management component accesses Kubernetes' API server to synchronize various asset attributes. This creates an automatic association between asset names and IP addresses, as well as enabling the real-time monitoring of changes to update the associations—which in turn provides essential information for traffic visualization, network micro-segmentation, and runtime security monitoring based on container assets.

Even though the Kubernetes service labels container assets during use, CloudArmour opts instead for a direct match between the asset name and its IP address, reducing communication overhead between security operators and service operators, and increasing efficiency.

Container Asset Traffic Visualization

The Kubernetes network orchestration capability for container pods lacks a segmentation mechanism akin to that of a public cloud VPC (virtual private cloud). Also, within the same cluster, different namespaces share a large Layer 2 network which results in interactions between containers in Kubernetes being trusted (all-pass) by default.

Controlling illegal access requires that it must first be detected. CloudArmour's secure container component uses the existing host system's traffic diversion mechanism to display traffic statistics between container assets, as well as traffic to external networks. With this capability administrators can easily visualize the interactions between container assets, detect illegal access, and provide a basis for controlling access between container assets using CloudArmour's micro-segmentation function.

Network Micro-Segmentation of Cloud Workload

A Kubernetes container network is essentially a large Layer 2 network, with standard east-west traffic between container assets in the environment. This architecture poses considerable risk to threats in the uncontrolled east-west traffic. Kubernetes supports the ability to control east-west traffic with NetworkPolicy when using the Calico CNI plugin, but this requires administrators to use YAML templates to orchestrate access control policies. The latter method is inefficient as it is not in line with traditional network security management practices.

To illustrate, consider an administrator with a clear understanding of the interactions between container assets. This administrator sets the access control policy globally to deny all, and configures an itemized access policy—such as allowing asset A to access asset B—using the NetworkPolicy YAML template. To achieve this, the administrator must write a YAML policy template for asset A that allows A's label to access asset B's label, as well as a YAML policy template for asset B to allow the reverse—a labor-intensive and unscalable process. NetworkPolicy has two additional drawbacks: it supports neither the configuration of explicit deny policies, nor any logging. Both these features are critical to a security administrator. At the same time, another popular CNI plugin, Flannel, does not support NetworkPolicy by default.

To overcome these challenges, CloudArmour utilizes the existing host system kernel's diversion mechanism to visualize asset traffic. At the same time, it provides a firewall function in line with traditional network security management practices. The firewall capability is realized entirely in the security container component, making it adaptable to any CNI mode. This construct also obviates the need to configure ACLs in the host's iptables to avoid coupling with the business administrator's local host's iptables policy.

An ideal network micro-segmentation solution defaults to least-privilege access rights, allowing only the minimal network access required by the workload. This can be extremely difficult to implement in a running business system. CloudArmour's Micro-segmentation Policy Assistant (MPA) feature directly addresses this need.

CloudArmour allows administrators to configure coarse-grained access policies based upon their familiarity with the business, and CloudArmour subsequently collects all session information for the current container environment using these policies. By manually selecting one of the policies, an administrator can view all sessions that have been allowed under that policy.

MPA also enables administrators to convert the collected session information into granular access policies with a single click. Leaving the granular access policies in place for a set time allows administrators to observe whether additional network access privileges are needed to fine-tune the micro-segmentation policies. Once no further sessions fall under the existing micro-segmentation policies, administrators can be confident that the fine-grained policies match all access relationships required by the business system. The resulting micro-segmentation policies govern the containers with least-privilege access rights, and the CloudArmour micro-segmentation solution can be implemented on the running business system.

Process Behavior Monitoring of Cloud Workload

CloudArmour's security guard leverages the existing messaging mechanism of the host system kernel to monitor, analyze and control the behavior of processes, network services, and file system access of container instances and nodes (hosts). To establish a baseline of security, CloudArmour uses machine learning to model the processes, network services, and file system access behaviors of container instances and nodes. Any behavior at variance with the baseline is monitored, alerted and controlled as exceptions.

It is technically possible to detect abnormal events such as reverse shells, escalated privileges, and escapes. Support for block-lists and allow-lists allows administrators to directly control identified illegal processes, network services, and file system access behaviors, or to ignore legitimate processes, network services, and file system access behavior.

Vulnerability Scanning for Cloud Workload Environment

Docker images are layered, starting with a base image then adding files, compiling, and running commands—layer by layer—onto the base. CloudArmour's security guard detects the base image version, installation package, executable, and other information through hierarchical parsing. It then compares the information gleaned with the standard CVE vulnerability database (synchronized from the cloud) and reports on different levels of vulnerabilities in the container images.

Vulnerability scanning for nodes operates in the same manner. CloudArmour provides vulnerability scanning for container images cached locally by the host, as well as for container images in public or private repositories. CloudArmour uses the Webhook message notification mechanism to automatically rescan updated container images.

To contain risk, CloudArmour can prevent container images—determined to have high-risk vulnerabilities—from running as container instances. A container image forms the basis of container functionality, and it typically has rapid version iteration in the software development process. Vulnerabilities are often introduced during this stage, and it is recommended that administrators configure regular vulnerability scanning of all container images. Expeditious remediation is also imperative, requiring the timely reporting of an image vulnerability situation to an administrator, and rapid repair of high-risk vulnerabilities.

Compliance Checks for Containers, Container Images, K8s, and Nodes

The container environment includes assets such as container applications, container images, K8s Master, K8s Worker Node, and other components. These assets are involved in varying degrees in data file permissions and parameter configurations—all of which must be checked for security compliance.

CloudArmour makes it easy and simple to implement the CIS security baseline in container environments. CloudArmour's security container component can retrieve all container configurations (application configuration can be obtained from YAML; container image configuration can be obtained from the Dockerfile; Docker/K8s/Linux-related configurations can be gleaned from the K8s Master or K8s Worker Node host), compare them to the CIS specification, and report on qualified or unqualified assets.

The Value of Hillstone's CWPP Solution

IT technology constantly evolves, necessitating security products and solutions to change in lockstep. Physical security appliances were popular in the era of physical servers in enterprise-owned data centers. Subsequently, security products have followed the era of virtualization to protect VM environments and similar cloud technologies. In the modern era of containers, security products must naturally also evolve their own container form.

Hillstone's container security solution, CloudArmour, provides enterprise IT teams with cloud-native container security capabilities for the Kubernetes environment. Capabilities include asset management, container asset traffic visualization, micro-segmentation for containers and nodes, monitoring of container and node process behavior, as well as vulnerability scanning for container images and nodes, and compliance checks for containers, container image, K8s, nodes, and other components.

Hillstone CloudArmour solution benefits include:

1) Agile: Cloud-native technology for easy and fast deployment.

2) Highly reliable: Cloud-native architecture resulting in a short fault detection path.

3) Highly efficient: Low compute resource requirements, and compatible with various CNI modes.

4) Easy to use: Easy to manage, with a graphical interface, self-synchronization of assets, and self-learning policies.

5) Zero interference: Easy control of enabled functions and low business interference.

Hillstone CloudArmour combines runtime protection and micro-segmentation to secure cloud-native applications and workloads against vulnerabilities and exploits. It integrates vulnerability management and compliance across the entire lifecycle of applications. It also provides deep visibility into cloud workloads with full security control, helping organizations meet the security demands of both the evolving software development environment, as well as the modern cloud container-based software architecture.

About Hillstone Networks

Hillstone Networks' innovative and accessible cybersecurity solutions reshape enterprise security, enabling cyber resilience. By providing enterprises with the visibility and intelligence to comprehensively see, thoroughly understand, and rapidly act against multilayer, multistage cyberthreats, Hillstone's products are favorably rated by leading analysts and trusted by over 23,000 global companies. With a reputation for "security that works," Hillstone's product suite covers enterprise edge to cloud and includes NGFW, SD-WAN, ZTNA, NDR, XDR, and CWPP. Hillstone's cutting-edge solutions leverage AI/ML and integrate seamlessly into SecOps frameworks, providing CISOs the assurance that their enterprises are well-protected while enabling a lower TCO.

Hillstone
NETWORKS

Visit **www.hillstonenet.com** to learn more

or contact Hillstone at **inquiry@hillstonenet.com**