SANS | GIAC CERTIFICATIONS

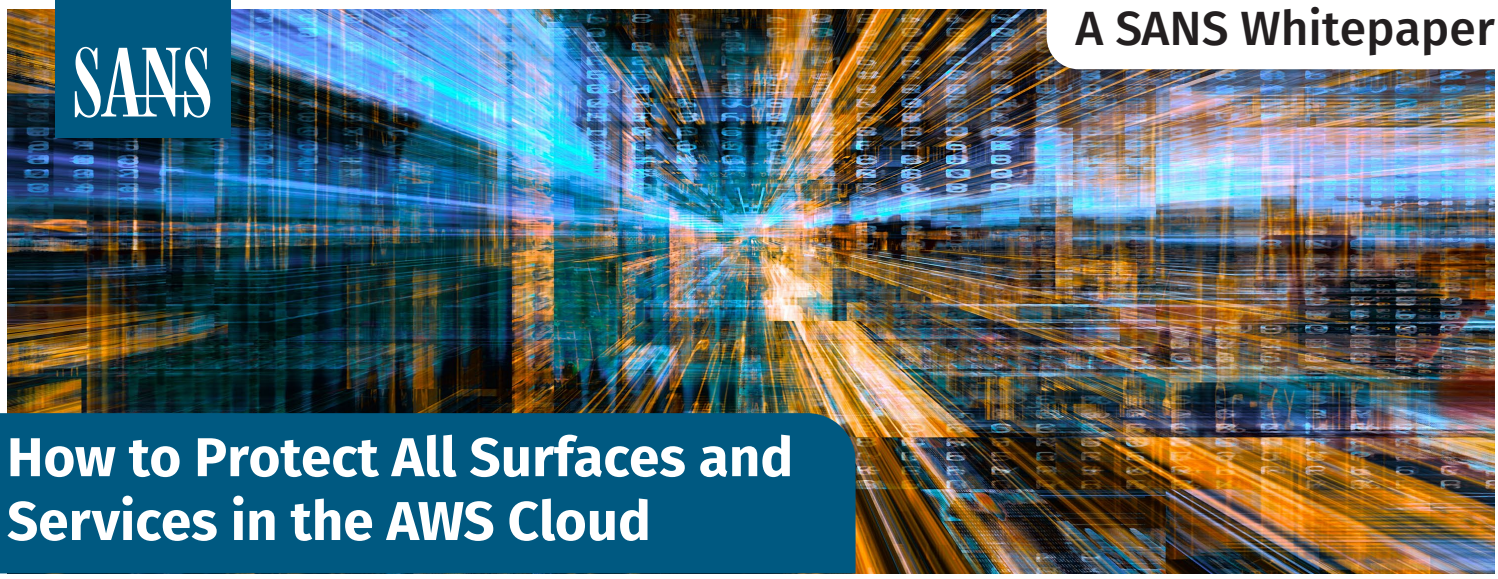
WHITE PAPER

How to Protect All Surfaces and Services in the AWS Cloud

Dave Shackleford

SANS

A SANS Whitepaper



How to Protect All Surfaces and Services in the AWS Cloud

Written by **Dave Shackleford**

July 2020

Sponsored by:

AWS Marketplace

The Dynamic, Interconnected Cloud Infrastructure

For some time, many organizations have been steadily increasing the scope of their cloud deployments. At the same time, large cloud service providers have expanded their catalogs of advanced cloud infrastructure and services, making it easier for a wide variety of IT and business teams to take advantage of cloud scale and capabilities. Today, it's common for more mature enterprises to make use of both platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS) capabilities in a unified cloud environment.

PaaS services are customized system and application platform stacks that vendors offer to customers on demand. These usually take the form of standardized OS platforms or OS platforms with very specific application stacks installed, and they are often used for application development or customized hosting requirements. In almost every PaaS implementation, the provider offers a standardized set of APIs that is extended to customers. These APIs are then built into applications developed and deployed within the PaaS infrastructure.

With IaaS, the service provider hosts a complete virtual workload network for customers. With this approach, providers can support multiple distinct virtual machines (VMs) or interconnected VMs with supporting virtual infrastructure. In many environments, providers can also offer truly "private" networks and VMs, where the consumer pays for a physically separate hypervisor server to avoid the security and performance risks inherent in multi-tenancy. With either approach to the IaaS model, consumers and providers share in the security responsibility. Consumers provision VMs and virtual

networks and must update and configure these components continuously. Providers support the hypervisor, underlying networks and management components. A key takeaway is this: Consumers are responsible for all patching and configuration in IaaS environments, just as in their own data centers. In fact, many IaaS deployments resemble traditional hosting, but with the addition of dynamic provisioning and virtualization for scalability and rapid resource availability.

As this breadth of deployed services and assets has grown, cloud providers have steadily enhanced computing capabilities and services to make the cloud ecosystem even more tightly coupled, offering a staggering array of innovative services and controls that can radically improve how IT infrastructure operates in a combined PaaS and IaaS cloud. With this expanded environment, however, organizations must address new security requirements and responsibilities.

In this paper we talk about containers and platform services, orchestration tools such as Kubernetes, and more advanced tools and services for protecting cloud workloads altogether. As the surface area of our cloud deployments grows, we need to evaluate cloud-native and third-party controls that can be natively embedded and automated as much as possible to keep up.

Protecting Cloud Workloads

Today, organizations have a much better understanding of how to deploy a sound stack of security controls and tools for securing cloud workloads. All organizations should consider the following best practices when running server instances and workloads in the cloud:

- **Ensure that configuration standards are in place for all of the various operating system and build varieties you want in the cloud.** These standards may align with the Center for Internet Security (CIS) benchmarks or other industry guidelines, but defining formal standards for the organization is a good start. Then, create VM image templates that have base standards in place and save them as cloud VM formats, such as Amazon Machine Images (AMIs).[1] Providers such as Amazon Web Services (AWS) have prebuilt images with many benchmark configuration controls already in place, too.
- **Implement a uniform tool, such as Chef[2] or Puppet, that you can use both in house and in the cloud to automatically configure VMs; then monitor the configurations in an ongoing fashion.** If you're already using Chef cookbooks or Ansible playbooks, for example, you can simply extend your configuration templates to cloud workloads. Categorize these templates assigned to workloads with tags or other metadata identifiers. Innovative cloud services such as AWS OpsWorks can dramatically simplify the extension of tools such as Chef into the cloud, enabling you to focus on configuration templates versus the operation of the underlying infrastructure.

[1] This paper mentions solution names to provide real-life examples of how cloud security tools can be used. The use of these examples is not an endorsement of any solution.

[2] Chef is a registered trademark of Chef Software, Inc.

- **If possible, patch and update only workload templates, instead of running systems; then deploy new instances and terminate the old ones.** This kind of progressive update cycle can take time to put in place, however, and goes hand-in-hand with a more progressive DevOps model of systems management and oversight.
- **Use cloud-native systems management tools, such as AWS Systems Manager, wherever possible.** This service offers a wide range of advanced capabilities for systems deployment and management, including package and patch management, remote administration and scripting control, scheduled tasks and jobs, system inventory and much more.

For more advanced endpoint security, look at cloud-friendly and cloud-native options first, if possible. Many endpoint detection and response (EDR) vendors have adapted their agents to be very lightweight and supported in all cloud platforms, and these are good choices. Anti-malware technology should be selected from the cloud provider marketplaces and integrated with all images stored in the cloud provider environment. Cloud-focused endpoint posture management tools should also be integrated with all images to ensure they're active for any new workload deployments.

As you get started with locking down cloud workloads, keep the following in mind to build a positive and productive feedback loop:

- Perform periodic reviews of the overall security posture within cloud environments to guarantee continued alignment of security and the other DevOps teams involved.
- Lock down system instances in the cloud as much as possible, commensurate with the exposure and data classification types involved.
- Pay careful attention to privilege allocation and user, group and role management associated with workloads. This can easily be overlooked over time in a dynamic environment.
- Commit to a culture of continuous monitoring, helping to automate detection and scripted response activities that minimize manual intervention wherever possible.
- Discuss vulnerabilities detected in cloud deployments with all team members. Involve DevOps teams in vulnerability, patch and configuration management discussions and policy creation.
- Discuss the changing threat landscape with DevOps teams. Solicit their feedback on practical measures that you can take to implement the most effective security without impeding progress or slowing down the pace of business activities.

While protecting workloads is always the goal, containers and serverless infrastructure may present some unique considerations that require new controls and solutions, as we discuss in the next section.

Containers and Serverless Infrastructure

For the most part, there are few changes to architecture for containers that differ from what you would have in place for VM environments. However, there are several changes to containers that you may want to consider:

- **Containers run on a shared OS platform with some form of container technology installed.** The OS itself is still potentially vulnerable, and ideally you should still install a HIDS/HIPS or anti-malware agent here, if possible. Given the focus on performance, however, consider lightweight technology, which is best suited for cloud and virtual infrastructure.
- **Containers require access to repositories to install and configure software packages.** You should perform due diligence with trust validation and security for repositories and software distribution, particularly in open source environments. Container security tools that focus on provisioning trusted and verifiable content should emphasize the following:
  - Survivable key compromise
  - Freshness guarantee
  - Configurable trust thresholds
  - Signing delegation
  - Use of existing distribution
  - Untrusted mirrors and transport
- **All container images should ideally be validated at various stages of the deployment pipeline.** Security of containers should include hardening (configuration), patching and monitoring. CIS has a set of excellent guides available for Docker.[3]

For security teams, the most important considerations to build into a process include:

- Defining a standard container image that is maintained
- Defining CIS Benchmark (or other) controls for container instances that need to be evaluated and met (permissions on files, logging, user privileges, etc.)
- Integrating container-specific container security platforms to securely validate images and repositories and monitor runtime behaviors

[3] Docker is a trademark or registered trademark of Docker, Inc.

Security Controls

For container-based security controls, implement any cloud-native container image scanning that your cloud provider makes available, such as the image scanning capabilities in Amazon Elastic Container Registry (ECR). These tools have native integration with the cloud provider environments and are often easy to automate for image check-in scans. For runtime security, however, you'll definitely want to investigate third-party tools from providers such as Aqua Security, Trend Micro, Snyk and Palo Alto Networks. Enable a virtual appliance or install a runtime agent into all container images for these tools to monitor runtime behavior and protect against bad actors.

Logging within containers and their underlying host OS should be considered where possible, and it's really important to make sure that no secrets are embedded in containers and container files. This is very common and requires security teams to scan container images and files for them continuously. (More on this in a moment.)

As DevOps and more dynamic microservices architectures become more prevalent in cloud deployments, engineering and architecture teams are using more lightweight cloud services to deploy application workloads. One of the more common technologies that many application development teams are employing is serverless, which offloads the entire workload (container and OS instance) to the provider's backplane, enabling developers to create microservices applications that only require application code to be uploaded and operated within the cloud provider environment. The increased use of serverless cloud infrastructure exposes some new potential concerns that security teams need to evaluate and address. For example, malicious code execution can be generated through serverless and stay resident in memory for extended periods of time.[4]

Implement any cloud-native container image scanning that your cloud provider makes available and immediately scan all new images for vulnerabilities.

While there may be potential code security issues in serverless (also known as functions-as-a-service, or FaaS) environments, the container host platform (OS, etc.) is largely out of scope, leaving the input, the code and the execution. For this reason, the first area organizations should focus on when developing a strategy for mitigating serverless security risks should be static code review. (Numerous third-party providers can integrate into serverless environments, such as AWS Lambda, to scan the code.) Many standard coding issues will apply regardless of where and how the code is run, resulting in the following core issues:

- **Event injection—**Injection flaws can still affect your code and how it handles the input. Better input validation and predefined database layer logic (stored procedures, for example) can help solve this.
- **Broken authentication—**Enforcement of strong authentication and role definitions for users of serverless apps should be emphasized.

[4] See the video "Gone in 60 Milliseconds," Rich Jones, https://c3subtitles.de/talk/673/

- **Insecure storing of app secrets—**Because API keys, encryption keys and other secrets are often involved in serverless function execution, security teams should ensure that developers are leveraging mature secrets management tools and cloud-specific key stores.
- **Improper exception handling—**Developers should ensure that errors and exceptions are handled appropriately, preventing stack traces from being exposed or displayed, etc.

Likewise, privilege and permission control over all serverless applications with cloud identity and access management (IAM) policies should be an area of emphasis, with organizations seeking to minimize the permissions that serverless functions run with, as well as the permissions of services accessing the serverless functions. Finally, every serverless API event (access, modification, execution and so on) should be logged within the native cloud environment with services such as AWS CloudTrail. Some additional areas of security related to configuration of serverless environments also need attention. Insecure deployment settings should be considered and mitigated by limiting memory usage and possible input vectors to serverless, and tuning execution timeout can help to mitigate DoS attacks and financial exhaustion.

Tools for Monitoring and Protection

Few native tools and controls are available today to protect serverless functions specifically. Security professionals should monitor all logs related to updates and changes to serverless function code and configuration. Teams should also ensure that the principle of least privilege is assigned to serverless code and that access is controlled to and from serverless functions within the cloud and from admins. Some providers, such as Aqua Security and Palo Alto Networks, can monitor and protect serverless functions from malware and remote execution attacks by monitoring runtime behavior, as well. Security teams may want to consider these tools if serverless functions are a key element of deployment models.

Organizations can integrate tools that provide more visibility into serverless execution, and others add security to the CI/CD stages and in-cloud execution (for example, container security tools such as Trend Micro Cloud One, Palo Alto Prisma Cloud [formerly Twistlock] and Aqua Security).[5] However, all of these tools require a significant dedication of resources to enable and tune.

[5] Cloud One and Trend Micro are trademarks or registered trademarks of Trend Micro Inc.; Prisma and Palo Alto Networks are trademarks or registered trademarks of Palo Alto Networks; Aqua Security is a trademark of Aqua Security Software Ltd.

Kubernetes and Container Orchestration Controls/Platforms

In the last several years, one container orchestration engine has risen to prominence above most others: Kubernetes. Kubernetes has a number of moving parts, and many security operations and engineering teams need more in-depth exposure to the platform so they can learn how to better protect it.

Deployment

As a starting consideration for Kubernetes security, you should consider whether you're managing the full Kubernetes management infrastructure or using a "managed" service in the public cloud. By that, we mean that many major public PaaS and IaaS providers have a Kubernetes service fabric that you can simply configure, and the providers manage the patching and upkeep of the systems. For example, AWS offers a managed Kubernetes service called Amazon Elastic Kubernetes Service (EKS). For many organizations, this could represent a simpler way to leverage Kubernetes PaaS deployments—you can even bring your own in-house Kubernetes configurations with you!

Many security professionals have described Kubernetes as an entirely new "cloud OS." This may or may not be an accurate description, but as a highly capable and granular orchestration and deployment engine for PaaS cloud infrastructure, Kubernetes is something that many security teams will likely be responsible for securing now and in the future.

A Kubernetes-based PaaS deployment has two major components:

- **Control plane—**First, there's the idea of a control plane. The *control plane* is the master controller (usually in the form of a master node) and includes API services, scheduling capabilities for containers and operational management tools/services. A master-level configuration database is also maintained in the control plane. In general, the control plane can be considered the brains of the Kubernetes infrastructure, and it needs to be very carefully protected.
- **Compute environment—**The second major piece of Kubernetes infrastructure is the compute environment, usually labeled *compute nodes* or *worker nodes*. These include communications services (the kube-proxy) and agents for master node control (kubelets), as well as pod configuration elements and container runtime services such as Docker. For stateful applications, persistent storage is likely to be included, and container images to spin up as part of the runtime engine operations will come from a defined container registry.

In the following sections, we break down the master node and worker nodes.

Master Node

With a Kubernetes master node, key Kubernetes components ensure that containers are deployed in a controlled manner and have adequate resources. The following components are important to understand:

- **kube-apiserver—**The Kubernetes API is the primary communications engine for the Kubernetes control plane, managing both internal and external requests. The API server is accessible via `REST` calls, with the kubectl CLI, or through tools such as kubeadm.

- **kube-scheduler—**The Kubernetes scheduler focuses on the overall health of the cluster and how new containers are handled and managed. The scheduler takes the overall pod resource consumption policy into account, as well as the cluster's current health. Based on these factors, pod deployments are scheduled for deployment to worker nodes.
- **kube-controller–manager—**The Kubernetes controller–manager is an overall master controller for all Kubernetes controllers in the cluster. A controller is responsible for the scheduler, which makes sure the pods running are healthy and the count of total pods is expected. Other controllers can be notified if pod configurations are somehow operating in an unexpected manner, and different controllers can manage the creation of accounts and API access tokens, as well. The master controller–manager is responsible for this coordination across the cluster environment.
- **etcd—**etcd is a simple key-value database that includes configuration data and overall data about the state of the cluster. In many ways, etcd is the master configuration for the entire cluster.

Worker Nodes

Worker nodes run the application components and containers needed to operate a PaaS infrastructure. Worker nodes are really the core workloads of a PaaS cloud infrastructure and may be full-blown instances, such as Amazon EC2 nodes in AWS, or fully managed infrastructure, such as AWS Fargate, that runs containers in a serverless fashion and doesn't require tenant oversight or management of the OS.

Security teams should understand the following key aspects of worker nodes:

- **Pods—**A *pod* is the core basis (or instance) of an application. Every pod is a definition of one or more containers that can operate in tandem with each other to provide application services.
- **Container runtime engine—**For containers to successfully run on worker nodes, a container runtime engine must be installed and operating. Docker is one of the most popular engines, but there are many other supported container runtimes, such as containerd and CRI-O.
- **Kubelet—***Kubelets* are small agents that communicate with the Kubernetes master node in a Kubernetes cluster. These small daemons are critical to ensure centralized operation and support are maintained in a Kubernetes cluster.
- **kube-proxy—**All worker nodes have a *kube-proxy*, which is a network proxy that enables Kubernetes networking services. Each worker node kube-proxy controls and manages node–node and external-cluster network communications for all nodes in the cluster. The kube-proxy can associate with the underlying OS network stack or use a software-defined network stack for container-container communications.

Pods are the foundational building blocks of container deployments with Kubernetes. An application that spans multiple containers is encapsulated in an object called a *pod* (although a single container could also be called a pod). Pods usually comprise multiple containers that operate together to form an application stack, have the same life cycle, and may need to be scheduled together on the same worker node(s). Pods are managed together and share OS, storage, and network address space. In production scenarios, it's common for a pod to have one or more containers running the primary application server and components with any number of secondary containers updating files and adding specific configuration updates as needed when code or packages change in a repository.

Kubernetes General Security Considerations

As with any software component, updating to the most recent version can significantly reduce risks associated with compromise of the system. Running unpatched software components with known vulnerabilities can easily lead to more ready exploitation, and exploiting a platform such as Kubernetes could bring severe ramifications because of its central control over resources. With a managed service such as Amazon EKS, patching is not the responsibility of the customer.

Defining roles and access controls for permissions to Kubernetes cluster and namespace resources is a critical aspect of locking down Kubernetes and PaaS deployments. In a managed service environment such as Amazon EKS, a wide range of prebuilt policies and roles that directly align with AWS Identity and Access Management (IAM) are available.

Logging in Kubernetes can be complex (handled in numerous places and not always simple to enable). Logs for each container can be sent to traditional files in **`/var/log`**, but Kubernetes best practices suggest *not* doing this because of how scattered this may be with ephemeral containers and performance issues. Instead, direct container logs to **`stdout`** and **`stderr`**; Kubernetes will aggregate them for analysis and access through **`kubectl logs`** commands. Amazon EKS control plane logging can also send audit logs from the Amazon EKS control plane directly to Amazon CloudWatch Logs. All API activity is logged in AWS CloudTrail, as well. Kubernetes API server audit logs are also interesting to security teams. Services such as Amazon EKS have numerous ways to collect and analyze these logs, using tools such as Prometheus (available through the Helm Kubernetes package manager utility).

Pod Security and Network Policies

Two additional elements that security teams need to consider are pod security policies and networking configurations. Pod security policies are applied at the cluster level to control pod configuration and operation. Pod security policies can control how (and if) pods are allowed to run. These policies can control a vast array of pod configuration options, including those shown in Table 1.

Table 1. Pod Configuration Options

Configuration Options	Description
Running of privileged containers	Allows privileged container contexts to be operated within pods.
Usage of host namespaces	Certain namespaces can be restricted if desired.
Usage of host networking and ports	Networking control for pods can be controlled at a policy level within the cluster.
Usage of volume types	Different storage volume types can be allowed or restricted.
Usage of the host filesystem	Allows containers and pods to access the underlying host OS and filesystem. In most public cloud scenarios, this is less likely when using services like Amazon EKS.
Requiring the use of a read-only root file system	A read-only file system can prevent the host OS from being compromised by a malicious container.
The user and group IDs of the container	Privilege assignments for containers can align with host OS user and group designations.
Restricting escalation to root privileges	For some operations on the host OS, privileges are required. These can be restricted at a pod policy level.

Network policies define which containers, pods and resources can communicate within the Kubernetes PaaS environment. There are several important things to note about this network behavior designation:

- Containers in the same pod use localhost and ports to define which containers can communicate with each other.
- Pods in the same node use a local eth0 device and a virtual node bridge. Each node has a virtual eth0 network interface and a virtual bridge that uses this interface to pass virtual/container traffic across it.
- Pods across nodes require cluster-level route table controls. Each cluster has a route table that is consulted for IP blocks when resources in one node need to communicate with another.
- The kube-proxy process maps communications between pods and services. For back-end services that need to communicate across nodes, the kube-proxy process can handle mapping of addresses and pod resources as needed for service communications.

Protecting the Cloud Control Plane

As organizations start using more cloud services and resources, they end up with a staggering variety of cloud administrative consoles and interfaces for which they're responsible. These are known collectively as the *cloud control plane*. The cloud control plane can encompass a wide variety of elements. The simplest is the cloud administrative console itself, which needs to be locked down as carefully as possible, with limited privileges and user access, multifactor authentication, etc. However, because the cloud is a software-defined infrastructure platform, there are also many other aspects of the cloud environment that may fall into this category. The first is the wide variety of APIs open and available in the cloud. This could be the command-line

interfaces for IaaS, such as the AWS Command Line Interface (CLI) or others associated with Kubernetes or other technologies. Another aspect of cloud control plane security is the question of the network zoning and segmentation put in place: What is exposed to various network environments?

Elements of Cloud Management

Cloud security issues can arise from a lack of oversight into what controls are in place, how they're configured, and what changes are made in cloud environments. Gartner has described a cloud management wheel that encompasses a wide array of security and cloud management elements. (See Table 2.)

Table 2. Elements of Cloud Management[6]

Element	Description
Provisioning and Orchestration	Workload provisioning and coordination
Cost Management and Resource Organization	Focused on cost management (and a complex topic in its own right)
Cloud Migration, Backup and Data Recovery	Shifting of resources *into* cloud and then building DR/BCP planning around this
Identity, Security and Compliance	The entire spectrum of security controls and requirements needed in the cloud, with some emphasis on identity management
Packaging and Delivery	DevOps and the pipeline to deliver code and applications into cloud environments
Monitoring and Analytics	Monitoring for both operational and security events and issues
Inventory and Classification	Continuously evaluating what resources are deployed in the environment, which is even more critical given the dynamic nature of cloud
Service Requests	Ticketing, help desk and troubleshooting, among other things

In this mix, three categories stand out for cloud control plane security: "Inventory and Classification," "Monitoring and Analytics" and, of course, "Identity, Security and Compliance." Governance and automation of all these are also paramount.

Coming back to the theme of configuration issues driving security challenges and incidents, the following set of factors tends to consistently drive the need for cloud security management and oversight:

- **The cloud is programmable.** It's very easy to make a configuration error in the cloud, and with little to no oversight, this is almost a guaranteed outcome. Once again, the cloud is a software platform built for application development and deployment, and this is a common situation.
- **"Cloud" is easily leading to a major sprawl in new technology implementation.** Many new technologies are available and only a click away for technologists. In most organizations, cloud use is not meticulously planned and just "happens" over time, which naturally leads to technology sprawl. As most security professionals know, this is challenging to contain.
- **The cloud is not the same as on-premises tools, technologies and services.** While the concepts may be similar in some cases, the cloud is its own software platform (and every cloud is unique).
- **Cloud inventory can be complex, because of more sprawl and a lack of monitoring.** While the cloud is a platform that enables much more flexibility in building and querying inventory of assets, it can be a challenge to achieve a continuous asset inventory without deep and highly embedded monitoring.

[6] Adapted from Gartner's cloud management wheel: https://blogs.gartner.com/marco-meinardi/2018/05/01/evaluating-cloud-management-platforms-tools-with-the-gartner-toolkit/

CSPM Solutions

In addition, privacy and regional requirements for security controls can make sound cloud security management even more challenging, as can implementation of multiple diverse cloud services.

CSPM tools and services can monitor for a wide variety of issues within any cloud environment they monitor. The idea is to create a policy on what the desired state or desired configuration is for the cloud infrastructure, then monitor the actual state of what is in place. Examples of cloud control plane issues CSPM can look for include:

- No encryption enabled for cloud storage or databases
- No encryption for traffic in sensitive data in motion
- Lack of sound key management (old keys, stale keys, etc.)
- IAM policies that don't adhere to least privilege principles
- Privileged accounts without multifactor authentication enabled
- Open or permissive network access controls
- Exposed data storage, such as accessible Amazon S3 buckets
- Minimal or no logging enabled within the cloud environment

Evolution of CSPM

In 2015, a category of cloud monitoring services labeled "Cloud Infrastructure Security Posture Assessment" (CISPA) was first proposed. This set of solutions centralized cloud visibility by scanning IaaS cloud interfaces, but the products and tools were highly immature and created a variety of false positives with overly complex user interfaces and weak reporting. Needless to say, this did not foster confidence in the space. Several years later, Gartner defined "cloud security posture management" (CSPM) as a group of security products and services that focused on compliance monitoring, dynamic cloud and DevOps integration, more thorough investigation and incident response capabilities, and risk assessment and reporting (that was much more accurate) for the cloud control plane.

When evaluating CSPM solutions, security teams should look for key features that a mature service offering should provide:

- **Configurable and automatable remediation capabilities—**Ideally, any discovered issues can be remediated automatically or with minimal manual intervention.
- **Custom policy and rules engine enforceable across a diverse cloud environment—**The granularity and flexibility of a policy engine is one of the most important features for any CSPM solution. These need to properly and accurately assess cloud service provider settings and asset configuration.
- **Integration with DevOps pipeline stages and tools—**For any code or image repositories, build tools, and so on, a CSPM platform should ideally be able to integrate and monitor activity here as well.
- **Detailed and configurable reporting—**Because CSPM is really a monitoring tool at heart, reporting is critical.

Applying CSPM to security operations should include the following:

- Asset inventory and classification—the faster and more accurate, the better
- Focus on identifying access to the cloud control plane
- Monitoring policies for configuration and compliance
- Monitoring operational policies and configuration (performance, etc.)
- Collecting artifacts and insight into incidents for incident response
- Visualization and reporting of control plane risks

Finally, security teams should consider the integration of CSPM solutions with their various cloud service providers. Most CSPM platforms are integrated through cloud service provider APIs and IAM service accounts, but security teams should closely evaluate the privileges needed and ensure any new IAM accounts are carefully set up and monitored.

Cloud-Native Solutions

There are also a number of cloud-native services that organizations can use for cloud security posture assessment. One great example of a service that can greatly enhance the automation of monitoring the state of assets and cloud control plane configuration in the cloud environment is AWS Config. AWS Config is a configuration monitoring toolkit for your AWS resources. It's possible to define baseline images, monitor systems continually, evaluate account settings and alert whenever a configuration changes. Another key feature of AWS Config is its inventory capability. In the cloud, all assets are software-defined and linked inextricably to the AWS backplane, so AWS Config can track them continuously. AWS Config can even perform automatic remediation, as well.

The Importance of Automation

With the significant increase in threat surface we now have, another key concept and skill set that security teams will need for cloud security enablement is automation. Information security will need to become a lot more automated and embedded into deployment processes to be successfully implemented in cloud operations now and in the future.

Security can scale with business operations in the cloud by tightly coupling to workloads. As a simple example, imagine a case where a business needs to rapidly increase the number of workload instances running in a cloud environment to support business initiatives. Ideally, the instance templates will already exist, and the operations teams involved would create new instances from those templates. If some form of host-based security isn't embedded into the template ahead of time, however, each new virtual machine will start in a less secure state. Even if a security agent is in place within the template, any changes to security policy would need to be rolled out to all new instances, which would slow down operations. Ideally, a flexible agent would fetch its policy when a new instance started, thus ensuring continuous and up-to-date security. With containers, embedded security tools and automated scanning and runtime security are even more critical.

Another example would be automated quarantine of a system where malicious behavior is detected. To facilitate network isolation, some type of automated detection and response cycle would need to be initiated based on indicators of compromise or unusual behavior that isolates the system at the host or virtual/cloud orchestration level, changing its network security groups designation to limit access. Disk and memory forensic acquisition could be automated, with the images copied automatically to a forensic node in the cloud environment, where encryption is automatically applied to protect them.

Network and vulnerability scans can be automated, too, continually sweeping cloud environments to determine the inventory of systems and services as well as their overall security posture. To monitor the cloud control plane, a tightly integrated set of CSPM tools and cloud-native services would likely prove most effective by monitoring active configuration, checking all API log events, and performing various reporting and remediation functions as well.

"Automation" has been a dirty word to security teams for far too long. Many security teams still think of all the possible things that could go wrong with automation, reminiscent of the days when all network problems were blamed on the firewall. Security teams need to familiarize themselves with orchestration tools and cloud APIs available to help implement prevention, detection and response tactics in the cloud.

Wrap-up: Security Innovation in the Age of Cloud

As the types of cloud services available grow and organizations begin to deploy large PaaS and IaaS environments that employ numerous interconnected services, the range of cloud security controls needed and surface to protect also gets larger. To keep up with the array of different cloud services, security teams should learn and use more advanced controls and develop more dynamic and continuous processes for evaluating security conditions in their environments. These might include:

- Cloud-native workload management and monitoring tools
- Template-based tools such as Chef, AWS OpsWorks, Ansible and more for configuration management and control
- Advanced image scanning and runtime security platforms for containers
- Embedded runtime monitoring tools for serverless functions
- Security platforms and services that natively integrate with leading orchestration tools such as Kubernetes and services such as Amazon EKS
- Cloud posture assessment tools for analyzing and remediating control plane security configurations

In all, these types of security controls and services are simply a natural evolution that reflect the nature of PaaS and IaaS software-defined cloud platforms and infrastructure. Security operations in large, distributed cloud environments need to adapt to accommodate more dynamic deployments and changes, new services and workloads, and a significantly greater reliance on automation.

About the Author

Dave Shackleford, a SANS analyst, senior instructor, course author, GIAC technical director and member of the board of directors for the SANS Technology Institute, is the founder and principal consultant with Voodoo Security. He has consulted with hundreds of organizations in the areas of security, regulatory compliance, and network architecture and engineering. A VMware vExpert, Dave has extensive experience designing and configuring secure virtualized infrastructures. He previously worked as chief security officer for Configuresoft and CTO for the Center for Internet Security. Dave currently helps lead the Atlanta chapter of the Cloud Security Alliance.

Sponsor

SANS would like to thank this paper's sponsor: