aqua

Practical Guide:

NIST Special Publication 800-190

Application Container Security Guide

Contents

Introduction

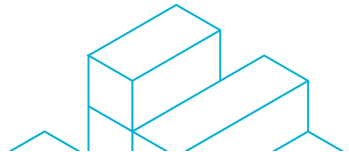
The NIST (National Institute of Standards and Technology, part of the U.S. Dept. of Commerce) has released a container security guide to provide practical recommendations for addressing the container environment's specific security challenges. This document covers the major risks and their security countermeasures organizations should consider deploying according to NIST. For more information and best practices, please **contact us.**

The Challenges According to NIST

Containers introduce dramatic changes into application development. They often drive an increase in the use of open-source components, and they also accelerate the pace of software development, challenging established security check-points to keep up. This new process may introduce vulnerabilities, and evade vetting processes based on existing version and configuration management.

Further, the container stack is radically new. Containers run on a shared kernel, which limits the level of isolation, and they require dynamic networking – both of which make it harder to have visibility and control over the runtime environment. This might also render existing, non-container-native countermeasures, such as IDS/IPS and firewalls, ineffective, in that they have no visibility into the activities of running containers.

This section offers a summary of the container environment challenges according to NIST.



3.1. Image Risks

As container images are distributed as is into hosts using orchestrators, you need to ensure the images are free from known vulnerabilities and misconfigurations. It is essential that developers and IT operations only use authorized images, which were scanned and available from a trusted registry.

Containerized applications cannot really be modified or patched once they are running. Even if you do, your orchestration will overwrite and reload that container from the registry image and not the patched image. Therefore, it's important to address known vulnerabilities and configuration errors at the development phase.

Image vulnerabilities and misconfiguration can be sensitive data, such as PII/GDPR in an image, embedded secrets, image that was configured with root privileges, and image with malware at the base image or known CVEs vulnerabilities.

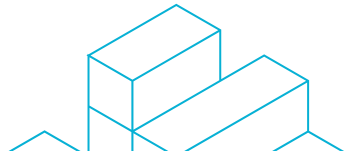
But that is not all. Oftentimes, vulnerabilities are discovered after the image was containerized and instantiated. If this were to happen, how would you handle it?
Scanning for known vulnerabilities in images and registries to ensure images deployed into production are free from known vulnerabilities, is a good starting point in your container security program. But vulnerabilities can be discovered long after containers are running and you need to know which container is using a vulnerable image/process. Therefore, it is important, as described in the following sections, to combine image scanning with runtime protection and detection capabilities.

3.2 Registry Risks

Many users, including developers, DevOps and auditors, access registries and other container resources at different stages in the pipeline. User access should therefore be secured and managed using a granular access control model that enforces least privilege policy at the image/registry and even container level while providing full accountability. User access privileges should be defined according to role, allowing or blocking specific actions such as write, view and more to prevent intentional or accidental damage to images.

Further, user access to registries should be via secure channels to prevent attackers from exploiting vulnerable images in addition to conducting 'man-in-the-middle' attacks to intercept intended connection registries and steal privileged credentials.

In a container/DevOps environment, the 'health' or stability of an image or app is measured against its lifetime, however, in a reverse manner. If in the past, app stability was measured by the number of months the app was running, for example, with images, it's just the opposite. Images are continuously scanned and replaced with new and updated images much more often (yes, as like with secrets, image rotation is a security best practice). Recycled images are more secured than



stale, unpatched images. To maintain trusted registries, images should be continuously scanned and recycled. In addition, images that were pulled from outside of the pipeline should be scanned as well, in case they will be used as base images.

3.3. Orchestrator Risks

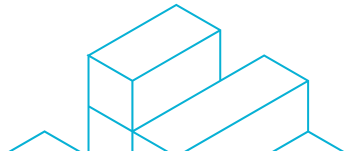
Access to orchestrators, as to any other sensitive asset, should be managed and controlled centrally and not separately from an organization's directory, as orchestrators run various apps with different sensitivity levels, that are managed by different teams. Organizations need to ensure that users granted access to orchestrators is limited to the needs of their specific job and based on an organization's access security policy. This can help organizations reduce the risk of exposure to abuse or error.

Another orchestrator related risk is regards networks. Traditional network monitoring solutions have a blind spot when it comes to network traffic between containers/nodes. This is due to the way the host environment is architected whereby the shared OS runs the container engine and the engine runs the containers themselves. Therefore, the OS is 'aware' of the container engine but is not 'aware' which containers are running. The container engine on the other hand, 'knows' which containers are running but is not aware of container activity. So, if you're running application firewalls or host-based intrusion prevention systems (HIPS) to monitor the OS, you'd be lacking both the visibility into container activity and the ability to monitor and control the same host containers' traffic. This is even risker where you have a mix of non-sensitive and sensitive container on the same node or virtual network, as the NIST states. For example, a public-facing app resides on the same node as your PCI-related app. As sensitive related apps are exposes to greater risks from a network attack, it would increase your attack surface.

In addition, NIST lists examples of consequences caused by weak orchestrator configuration, including unauthorized host added to a cluster, comprised shared key of clusters that can compromise all nodes, and unencrypted, unauthenticated privileged user communication /access to an orchestrator.

3.4 Container Risks

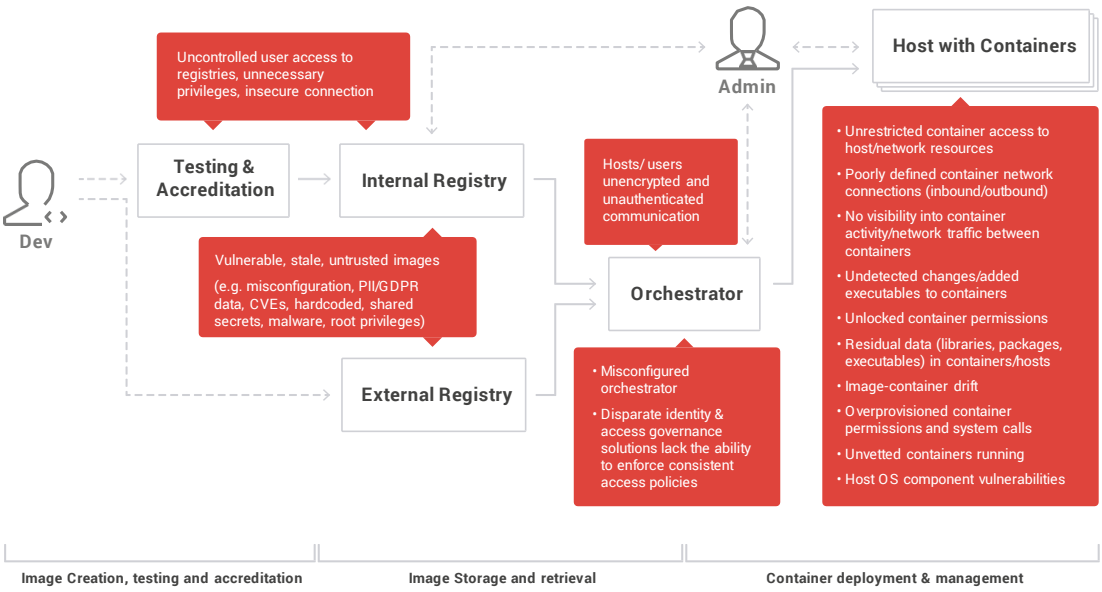
Malicious activity between containers/host OS can be originated by malicious software and expand to other resources due to poorly defined outbound/inbound connections. Any outbound connectivity is a potential attack path, so a good best practice would be to limit ingress and egress points to the necessary limit. As noted above, no visibility into containers' activity can lead to compromised runtime environment. In addition, NIST details runtime misconfigurations as overprovisioned sys calls, unneeded executables, or container privileges (e.g. running as root) which enable the container to act as part of the host OS or make changes to host files, location, etc. and access all other containers on the host.



The challenge is not only how to prevent these types of risks but in doing so without blocking legitimate access to services required by the applications. Multiply this by a large and proliferating number of applications and container types and the task becomes ever more challenging. Therefore, communication rules, access management, secrets management, and permissions, should be automatically defined in the container level, not on the host or cluster level.

In addition, rogue containers can be originated from an image drift, once an image is instantiated or while in runtime. Even if images are scanned and authorized in the build environment, they can be changed maliciously or unintentionally in runtime. Further, scanning containers in runtime is a *too much too late* practice. It's not only a waste of host resources but also ineffective as it requires the image to go back into pre-production for remediation instead of it being stopped from being instantiated in the first place. In cases where images are already instantiated and deploy into runtime, any patching of running containers is also ineffective, as mentioned above, as your orchestration will overwrite and reload that container from the image and not from the 'patched' container.

Container Threat Landscape



NIST SP 800-190: Major Risks for Core Components of Container Technologies

3.5 Host OS Risks

Vulnerabilities in hosts can be caused by overprovisioned user access rights, unnecessarily bloated OS resources, unpatched/not up to date OS and container engine versions, open network ports, badly configured authentication, and lack of namespace isolation.

Finally, in the absence of a systemic, container-native approach to visibility and control for the container layer and host-level activity, containerization can be a minefield.

It is therefore important to combine image and host scanning with runtime protection and detection capabilities. The combination of image scanning and assurance policies with running container behavior analysis and network rules enforcement, enables organizations to keep their pipeline and production environments safe. By gaining insight on the image content as well as the container's expected activities, controlling container communication and user access based on role and permissions, incident response teams can quickly detect and respond to unauthorized activities.

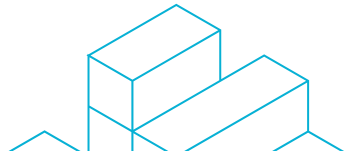
NIST Checklist for Container Security

A practical guide to help organizations achieve and demonstrate compliance in the container environment

The following checklist provides a summary of key recommendations that NIST introduces as well as important actions organizations should take to help achieve and demonstrate compliance. Aqua's cloud-native solution enables organizations to implement most of the countermeasures presented by NIST. If you have any questions or which to schedule a demo, **contact us**.

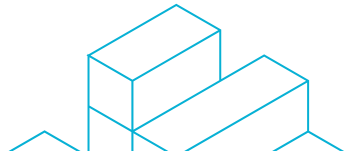
NIST Recommendations	Aqua Feature Addressing the Requirement
4.1 Image Countermeasures	
4.1.1 Image Vulnerabilities. Use container-native vulnerability management solution which 1. Integrates into the pipeline and the image	- Aqua provides an inventory of containerized applications, covering the different repositories, images, containers, and hosts in the organization - Aqua vulnerability scanning is integrated into the build process, enabling developers to seamlessly perform vulnerability/configuration scans and apply remediation at the build phase using native build tools (e.g. Jenkins, TeamCity, etc.)

lifecycle; from image build, through registries to runtime 2. Provides visibility into vulnerabilities at all layers of the image, cross all apps, enabling reporting and monitoring capabilities 3. Offers policy-driven enforcement to ensure image compliance by establishing “quality gates” at each stage of the build and deployment process	- Aqua connects to image registries and enumerate all images stored in them. For each image, it creates an inventory of the installed packages. Aqua will also process all images stored on the hosts, that were not pulled from an image registry, and create a package inventory for every image - Aqua uses multiple resource feeds for scans (public CVEs, vendor-issued, proprietary vulnerability data streams and malware) to achieve refined results and less false positives - Aqua enables compliance and security users to implement their own custom compliance checks - Image scans are ranked based on severity and CVSS scores. CVSS score is included in risk results for an image and can be used as policy criteria for image acceptance - Each image is scanned for vulnerabilities both in its OS packages and development language files - Aqua provides image bill of materials, lists all image packages files and layer history - Aqua provides validated chain of custody and distinguishes between vulnerabilities that were added to the code by the developer and known vulnerabilities in the base image for fast and effective remediation process - Aqua enables “quality gates’ throughout CI pipeline using distinct image assurance policies. Policies are automatically allowed/disallowed images based on set of criteria such as usage context, image name, label and registries - With Aqua image assurance policy, users can block images that have not been vetted by Aqua (e.g. images that were not scanned), meaning that only approved images will be allowed to run - A developed image will be promoted to production only if it passes all the required tests. Aqua admin can also assign labels to images and create security policy that allows only images with specific labels to enter production - Aqua provides actionable mitigation information on detected vulnerabilities for fast and effective remediation

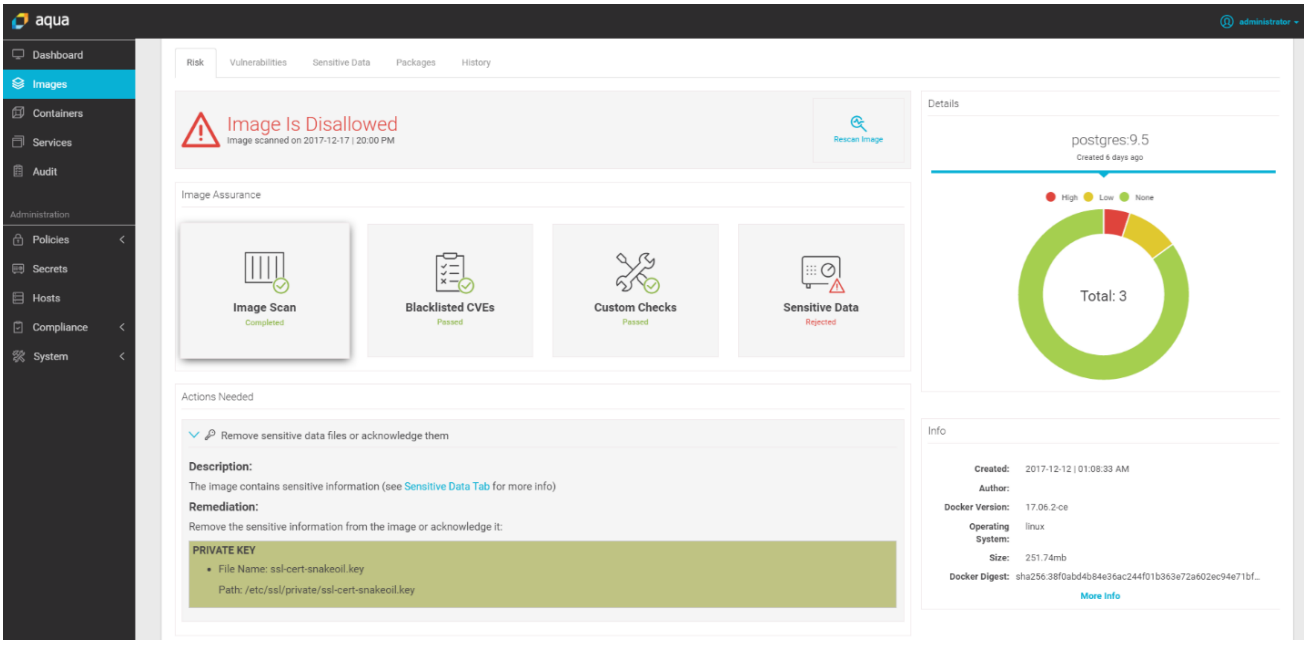


4.1.2 Image configuration defects. Have processes in place to validate and enforce image compliance;
 	
1. Validate image configuration settings
2. Consensually monitor image compliance state
3. Block non-compliant images from running
4. Use trusted sources for base images and continuously update base layers to reduce attack surface
5. Control user access to runtime containers
	- Aqua continuously scans image OS packages and development language files for known vulnerabilities and misconfiguration immediately after build, as the image is pushed to a registry, to:
	  ○ Identify the base image used for the build (e.g. is the image from a reputable source?)
	  ○ Identify vulnerabilities in open source packages
	  ○ Identify if image is configured as root/admin
	- Images that do not pass acceptance criteria are marked as 'Disallowed' and will not be run on nodes protected by the Aqua Enforcer
	- Aqua admin can upload custom compliance checks
	- Aqua provides broad set of predefined image assurance policies
	- Auditor can track changes in vulnerability status and maintain vulnerability vs. remediation trends
	- On a production node, Aqua allows to pull/run only images that are:
	  ○ Registered and their risk posture is known (use unique identifier for images, resilient to name changes)
	  ○ Passed image risk policy or have been explicitly allowed by auditor
	  ○ Block unauthorized images (e.g. blacklisted images) from being deployed and instantiated in production
	  ○ Prevent changes to executables and adding new executable to file system once containers are instantiated
	  ○ Enforce container least privileges
	  ○ Remove unused/unnecessary executables
	  ○ Identify and block unregistered images running in production

4.1.3 Embedded malware. Continuously monitor images for embedded malware, including malware signature sets and behavioral detection heuristics method.	- Aqua scans images for malware as part of the image assurance policy - Aqua enables zero-configuration container behavioral profile to establish a baseline of behavior. This behavior is enforced on a granular level with the stated criteria. - Aqua enforces image immutability by not allowing packages and components not present in the original image to be introduced to an existing image or container - Aqua logs all access, Docker commands, container activity, secrets usage and system events
4.1.4 Embedded clear text secrets. Securely store secrets outside the image and deliver them only to authorized containers on demand. Encrypt secrets at rest and in transit.	- Aqua scans for hardcoded secrets (e.g. SSH keys) within images. After running a scan, Aqua generates an inventory view of secrets, their locations, who/what has access to them and risk score - Aqua provides central management and secure distribution of secrets and keys into running containers, as well as updates/rotation/revocation with no container downtime - Secrets are not visible outside the container and encrypted in transit - Secrets are not persistent on disk on the host - Aqua integrates with CyberArk EPV, CyberArk Conjur, HashiCorp Vault, AWS KMS and Azure Key Vault as the centralized secrets stores - Aqua monitors container secrets activity, tracks and logs secrets usage

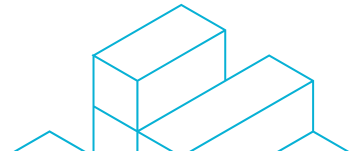


4.1.5 Use of untrusted images. Maintain set of trusted images and registries by
• Continuously scanning for vulnerabilities and misconfigurations
• Controlling authorized image and registries
• Maintaining and validating image hash to ensure only authorized images are running
	- Aqua scans images on any registry across any platform
	- Aqua uses a cryptographic digest of the image content across all layers to create a unique digital fingerprint. Images that don't match known fingerprints will be blocked from running, enforcing image integrity and preventing drift.
	- Aqua blocks an image according to any of the following:
	  ○ Risk score above a given threshold
	  ○ High severity vulnerabilities
	  ○ CVE blacklist
	  ○ OSS license types blacklist
	  ○ Malware
	  ○ Embedded secrets
	  ○ Not passing a custom compliance check or SCAP script
	- Aqua re-validates image status (allowed/disallowed) before instantiation
	- Aqua detects any changes to containers (binaries, certificate, hash, system calls)
	- Aqua associates containers to source code for end-to-end vulnerability visibility and traceability



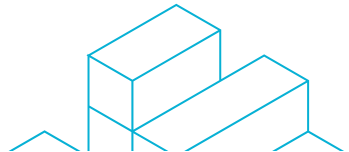
4.2 Registry Countermeasures	
4.2.1 Insecure connections to registries. Ensure all connection channels to registries are encrypted and all data pushed to and pulled from a registry occurs between trusted endpoints and is encrypted in transit.	- Aqua admin can limit DevOps users to pull and run images only from trusted registries, ensuring that only secured registries with encrypted channels are used.
4.2.2 Stale images in registries. Ensure only up-to-date, authorized images are used based on clear naming convention.	- Aqua allows daily scans of images to alert on out-of-date vulnerable packages, base-images and versions - Aqua allows the admin to define stale images via custom checks and block them from running - Aqua's labels feature allows categorization of images to differentiate between (for example) production and non-production images, or between different applications.
4.2.3 Insufficient authentication and authorization restrictions. Controls and manages user access to registries via integration with directory services. Audit and log access to registries (write access and read of sensitive data). Integrates automated scan into CI processes to ensure only authorized images can be used	- Aqua provides Role Based Access Control (RBAC) to limit super-user permissions, tasks and container resources (e.g. images, containers, nodes, networks, pods, volumes etc.) - Aqua provides both preconfigured and custom roles, that enable defining granular privileges on specific groups of containers, images, pods, hosts, etc. - Enforce effective security hygiene into the CI/CD build process via scan plugins to any CI tool (e.g. Jenkins, Microsoft VSTS, Bamboo, GoCD, Gitlab, TeamCity, etc.) - Aqua monitors and alerts on unauthorized user activity - Aqua logs all access activity for investigations and regulatory compliance
4.3 Orchestrator Countermeasures	

4.3.1 Unbounded administrative access. Orchestrators should use a least privilege access model to enable controlled and limited user access to sensitive resources (host, containers, and images).	- Aqua provides visibility into orchestrator configuration of these settings as part of the Docker and Kubernetes CIS benchmark checks (daily)
	- Aqua also provides its own controls to limit such access at the host, image and container level, regardless of the orchestrator used. With Aqua, admins can:
	  ○ Control which users have access to which K8s commands. For example, creates fine-grained user roles that govern access to *kubectl* commands, or assigned to specific *deployments* and *nodes*
	  ○ Using Aqua RBAC, admin can block users' direct access to host while enabling host management via the orchestrator



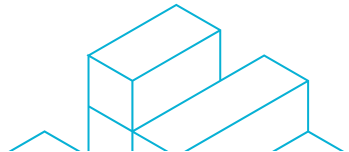
4.3.2 Unauthorized access. Use strong authentication methods (e.g. SSO) to secure access to cluster-wide admin accounts. In addition, encrypt data at rest and control access to the data from containers only, regardless of the node they're running on.	- Aqua admin can set and enforce user access policies to container resources (e.g. view, edit) - Aqua monitors user access, blocks and alerts on any unauthorized access attempts - Aqua's labels-based management can be used to label hosts as dev/test/production, and different RBAC and container policies can be applied by these labels, so that specific users can only access resources with specific labels - Aqua admin can set up policy to block and alert on any unauthorized connection to/from containers (e.g. file access) - Aqua continuously logs user activity including access and attempted access events

4.3.3 Poorly separated inter-container network traffic. Configure orchestrators to segment network traffic into discrete virtual networks by sensitive level (e.g. public-facing apps can share a virtual network vs. internal apps)
 	
4.3.4 Mixing of workload sensitivity levels. Configure orchestrators to separate container and hosting zones by automatically grouping and deploying workloads to hosts based on their sensitivity level, purpose and threat posture. Further, for additional layer of security, it's recommended to segment network traffic more discretely based on sensitivity levels as well.
	- Aqua's Container Firewall (Nano Segmentation) limits network connectivity between workloads by applying a firewall-like concept for the container environment. This capability allows creating network boundaries across services, where admins can control which networks are accessible for each service. Aqua admin can define
	  o Which containers' inbound/outbound ports are accessible to/from which IPs
	  o Container network connections based on set of service-oriented firewall rules, regardless of where container resides
	- Aqua's label-based management allows to label groups of containers as, for example, PCI/PII-sensitive, either within the Aqua console or by inheriting security group definitions from orchestrators.
	- With Aqua labels users can easily create a label based rule for operations or inventory purposes, for example, container with a certain label 'PCI-DSS compliance', will be blocked from having outbound connections, so any outbound connectivity will be automatically denied. Aqua automatically alerts and blocks unauthorized communication flows
	- Use labels and services to automatically group containers to be deployed on separate nodes and network segments.

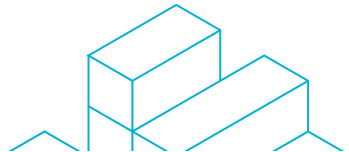


4.3.5 Orchestrator node trust. Configure orchestrators to safeguard secure-by-default ensuring nodes have a persistent identity, gain accurate inventory of nodes and their network connections. Have the means in place to isolate/remove compromised nodes would not compromise others and finally, use authenticated network connections between cluster members and end-to-end encryption of intra-cluster traffic.	- Aqua automatically discovers all image repositories, running containers, nodes, and hosts across the environment, mapping out network traffic, to enable admins to maintain up-to-date inventory of the container environment - Container network traffic is mapped across application services, regardless of actual network infrastructure used, and including networking within and across hosts - Aqua prevents orchestrators such as Kubernetes from deploying untrusted images Aqua performs host integrity checks, including vulnerability scan, malware and CIS test to ensure nodes are secured.

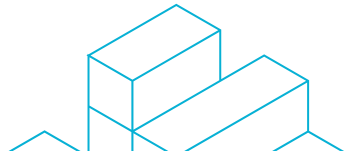
4.4 Container Countermeasures	
4.4.1 Vulnerabilities within the runtime software. Monitor container runtime for vulnerabilities. Use tools to detect CVEs vulnerabilities and ensure orchestrators only allow deployments to properly maintained runtimes	- Aqua provides a view of all running containers and their originating images, including package inventory for every running container - Aqua uses its image assurance fingerprinting to track containers back to their original images – by preventing drift from the original images, Aqua enforces the CVE risk policy and does not allow containers that were from unauthorized image to run. - Aqua continuously updates its cyber intelligence feed, to ensure that if a newly discovered vulnerability exists in a running container, this will generate an alert and allow remediation. - Aqua runtime protection also blocks unauthorized executables from running with no container downtime



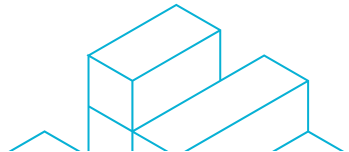
4.4.2 Unbounded network access from containers. Control and monitor containers' outbound network traffic as well as inter-container traffic. Use app-aware tools to gain visibility into inter-container traffic as well as to dynamically generate rules used to filter traffic based on specific app characteristics. In addition, these tools should provide: • Automated container networking surfaces (both inbound ports and process-port bindings) • Detection of traffic flows both between containers and other network entities • Detection of network anomalies (e.g. unexpected traffic flows, port scanning, outbound access to potentially risky destination)	• Aqua discovers and maintains up-to-date inventory of containerized applications, image repositories, and hosts across the cloud environment • Aqua maps a container network traffic across application services, regardless of actual network infrastructure used, and including networking within and across hosts • Aqua's Container Firewall (Nano Segmentation) enables admins to limit network connectivity between services by applying a firewall-like concept for the container environment. This capability allows creating network boundaries across services, where admins can control which networks are accessible for each service • Aqua has specific threat mitigation defenses to detect and prevent port scanning • Aqua has specific threat mitigation defenses to detect and prevent connections to IP addresses with poor reputation • Aqua admins can also deny inbound and/or outbound communications from/to containers in the network section when creating a new runtime profile for container • Aqua admins can set up alerts and preventive actions on a container's unauthorized activity including unauthorized network connections (inbound and outbound) • Aqua monitors and logs resources activity and consumption including network connectivity • Aqua admin can set real-time audit events on policy violations and send it to organization's SIEM solution. In addition.
4.4.3 Insecure container runtime configurations. Use tools/processes to continuously asses and automatically enforce configuration settings against CIS standards, for example. In addition, as an added control, consider using Mandatory Access Control (MAC) technologies to secure the host OS layer and ensure only specific files, path, processes, and network sockets are accessible to containerized apps. SECCOMP and custom container profiles also can also be used to constrain system-level and other capabilities where runtime containers are allocated. For high-risk apps, consider using additional profiles.	- Aqua enables the scan of container engine compliance against Center for Internet Security (CIS) benchmarks for both Docker and Kubernetes, and run those scans daily - Aqua controls and monitors container access rights to OS and host resources - Aqua prevents kernel operation (e.g. does not allow CHOWN) - Aqua centrally manages and enforces SECCOMP profiles - Aqua admins can set and control container memory and CPU consumption and running process limits to prevent DOS attack



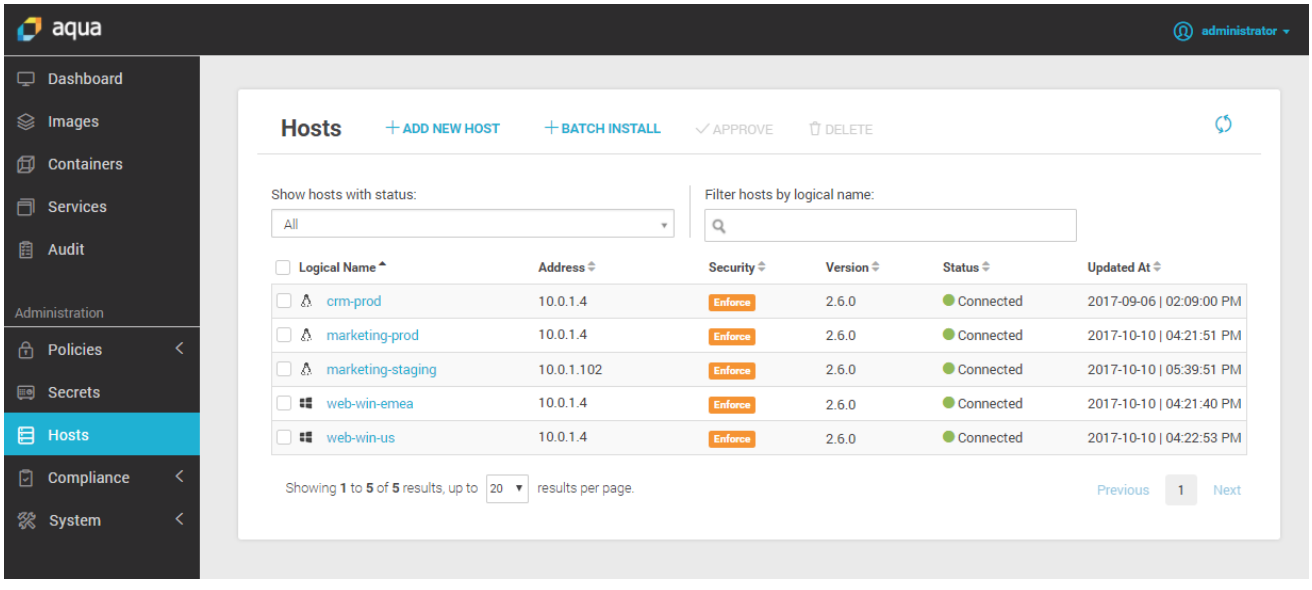
4.4.4 App vulnerabilities. Use container-native tools to automatically profile containerized apps using behavioral analysis and build security profiles to be able to detect and prevent anomalies event at runtime, such as: • Invalid or unexpected process execution • Invalid or unexpected system calls • Changes to protected configuration files and binaries • Writes to unexpected locations and file types • Creation of unexpected network listeners • Traffic sent to unexpected network destinations • Malware storage or execution Further, containers should also be run with their root filesystems in read-only mode to make the containers more resilient to compromise. In addition, writes privileges can be defined and monitored separately.	- Aqua automatically creates a security profile based on container behavior - file access, resource usage, read-only root filesystem, network settings, namespace settings, seccomp profile, and executables, baselining all legitimate container activity and using machine learning to create a whitelisting policy. This creates least privilege security runtime profile that Aqua admin can then edit and save - Aqua admins can also manually tweak the profile parameters for specific parameters Aqua detects and alerts on anomalies, if occurred. Only legitimate container activity will be permitted, automatically preventing many types of malicious behavior and privilege abuse in real-time - Aqua scans images and hosts for malware - Aqua's security research team continuously provides threat mitigation protections ("IPS for containers") that block specific behaviors that are indicative of attacks, such as fork bombs and attempts to access malicious IP addresses - All events are logged by Aqua. Event details include the user, image name, container name, rule, and the reason for the event severity - Admins can also configure forwarding of events to external SIEM and analytics tools, such as Splunk, ArcSight, SumoLogic and more
4.4.5 Rogue containers. Create separate environments for development, test, production, and other scenarios, each with specific controls to provide RBAC for container deployment and management activities. In addition, container creation should be associated with individual user identities and logged to provide activity audit trail. Further, it is recommended to enforce baseline requirements for vulnerability management and compliance prior to deployment.	- Aqua's labels-based management can be used to segregate resources/environments by labeling hosts as dev/test/production, and different RBAC and container policies can be applied by these labels, so that specific users can only access resources with specific labels - By default, when admins deploy Aqua Enforcer on a container host, the Aqua Enforcer applies "owner-based access control". This means that a user who is a container owner (a user who created and started the container) has full container access, but other users will not have the same degree of container access. The default "owner-based access control" behavior might be applicable for most common container use-cases. To extend or reduce privileges admins can create policies and explicitly assign user permissions - Aqua integrates with the organization's identity management systems to map container users to the organizational user groups, and SAML for single sign-on. This allows even more granular access control and separation of duties



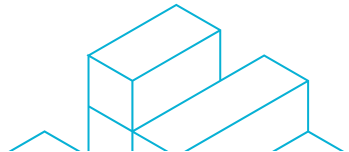
4.5 Host OS Countermeasures	
4.5.1 Large attack surface. Use container specific OS whenever possible, or follow NIST SP 800-123 to host-OS hardening practices (e.g. host that runs containers cannot run other apps/unnecessary system services, etc. to minimize attack surface). Further, continuously scan host (e.g. kernel) for vulnerabilities and updates.	- Aqua continuously scans host for vulnerabilities and malware - The host compliance configuration scan is done against the Center for Internet Security (CIS) benchmarks (Docker, K8s) - Restrict and enforce container's access to the specific resources it required
4.5.2 Shared kernel. Do not mix containerized and non-containerized workloads on the same host instance. (e.g. if a host is running a web server container, it should not also run a web server as a regularly installed component directly within the host OS). This will also make it easy to apply optimized countermeasures for container protection.	Restrict and enforce container's access to the specific resources it required
4.5.3 Host OS component vulnerabilities. Implement management practices and tools to validate the versioning of components provided for base OS management and functionality. Further, redeploy OS instances/apply updates (security and components wise), to keep the OS up-to-date. Moreover, to reduce the attach surface, the host OSs should operate in and immutable manner with no data as well as app-level dependencies uniquely stored on the host. All app components and dependencies should be packaged into containers. This will help in detection of anomalies and configuration drift.	- Aqua scans the host OS for vulnerabilities and malware - Aqua logs user login and logout events on the host - Aqua scans the host for configuration issues per the CIS Docker Benchmark



4.5.4 Improper user access rights. Ensure all authentication to the OS is audited, as well as monitor and login anomalies and privileges escalation to be able identify, for example, anomalous access patterns to host and privileged commands to manipulate containers.	- Aqua audits all login attempts to the host, including invocation of sudo programs - Aqua controls all commands that can manipulate containers; such attempts are audited and allowed/denied based on user access control rules
4.5.5 Host file system tampering. Ensure containers are running with minimal set of file system permissions required. Very rarely should containers mount local file systems on a host. Instead, any file changes that containers need to persist to disk should be made within storage volumes specifically allocated for this purpose. In no case should containers be able to mount sensitive directories on a host's file system, especially those containing configuration settings for the operating system. Use tools to monitor which directories are mounted by containers and prevent these containers form being deployed.	- Aqua automatically creates an image security profile based on container activity, which Aqua tracks and analyzes. The profiler analyzes and reports on the security profile by noting vital component usage, resources, and network settings. This creates a least privilege security runtime profile which Aqua admin can then edit and save - Aqua enables admins to restrict containers from specific mounting volumes or from writing into specific volumes or directories - Any behavior not allowed by the Aqua container behavioral profiles will result in alerts being generated and logged - Aqua records all access, Docker commands, container activity, and system events and provides a full audit trail



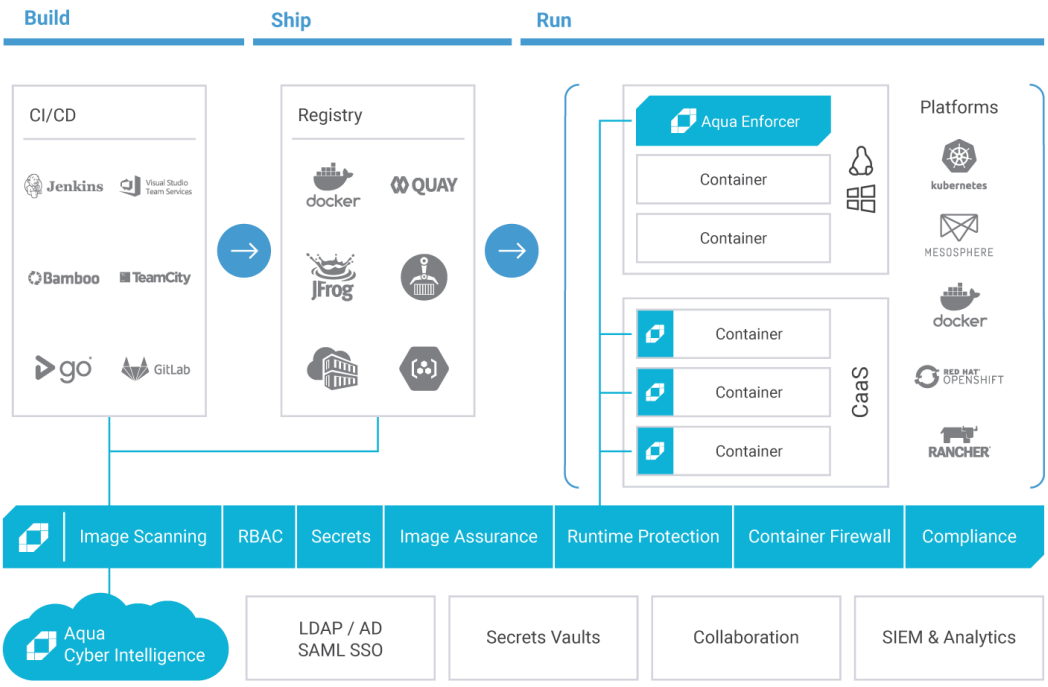
4.6 Hardware Countermeasures. Establish hardware/firmware root of trust to ensure containers are being run in a secure environment, by: 1. Measure firmware, SW, and configuration data before it is executed using a Root of Trust for Measurement (RTM). 2. Store measurements in a hardware root of trust, such as trusted platform module (TPM). 3. Validate that the current measurements match the expected measurements. can be trusted to behave as expected. Extend the chain of trust to the bootloaders, OS kernel, and the OS components to enable cryptographic verification of boot mechanisms, system images, container runtimes, and container images. For container technologies, these techniques are currently applicable at the hardware, hypervisor, and host OS layers, with early work in progress to apply these to container-specific components.	- Aqua recommends that customers use TPM tools for HW countermeasures, where applicable (this requirement may not be relevant to certain cloud or multi-tenancy deployments).



Aqua Container Security Platform: Quick Overview

Aqua's platform is a container-native, full lifecycle solution for securing container-based applications. The Aqua CSP comprises three main components:

1. **Aqua Command Center:** A central management component can be deployed on multiple instances for high availability. It provides policy management, image scanning, image lifecycle controls, monitoring, and reporting. It integrates with image registries for image scanning, with CI/CD tools for security testing as part of the build, and with SIEM/analytics to output audit and alert data. The Command Center exposes full API access as well as a management console UI.
2. **Aqua Enforcer:** The Aqua Enforcer is itself deployed as a container, in charge of monitoring and controlling container operational actions on every protected node. It has visibility into the activity of other containers on the node, and employs multiple methods to stop specific activities that do not comply with policy.
   The Enforcer communicates the event stream back to the Command Center, which in turn publishes the policy out to the Enforcers. All communications are encrypted using SSH and mutual authentication protocols to prevent spoofing.
3. **Aqua MicroEnforcer:** Protects container-as-a-service (CaaS) deployments where there's no host/cluster to manage, such as AWS Fargate and Azure Container Instances - Aqua embeds its security controls into the image during build.
4. **Aqua Cyber Intelligence:** Cloud-based service that supplies container vulnerability, malware and threat intelligence to Aqua deployments. The service relies on multiple public and proprietary sources, and provides "virtual patching" for several attack vectors, all of which are updated regularly.



* * *

For more information on the Aqua Container Security Platform,
or to schedule a product demo,
contact us.

Aqua enables enterprises to secure their container and cloud-native applications from development to production, accelerating container adoption and bridging the gap between DevOps and IT security.

The Aqua Container Security Platform provides full visibility into container activity, allowing organizations to detect and prevent suspicious activity and attacks, providing transparent, automated security while helping to enforce policy and simplify regulatory compliance.

For more information, visit www.aquasec.com