Securing the container platform

Build a chain of trust

The DevOps challenge: Securely innovating at velocity

Supporting business objectives in highly competitive markets requires application development executives and their teams to deliver high-quality customer experiences across device types at an accelerated pace. As a result, DevOps teams increasingly use container-based cloud platforms with agile collaboration methods and a toolchain for maximizing automation in the process of creating and iterating cloud native apps as independent but interoperable microservices.

While setting up and maintaining excellent security would seem to add friction when moving toward a cloud-based DevOps scenario, security definitely cannot be ignored. Most cloud platforms use Docker for containers, for example, and containers run on a shared Linux kernel, inheriting its security challenges. Downloaded from a community exchange, undetected rogue container software that obtains a privileged escalation on one host's Linux kernel could begin exfiltrating data or branch out to do damage through denial of service (DoS) attacks. Containers can also interfere with other containers because they all share access to resources such as channels, libraries and binaries.

With greater adoption of the bring-your-own-device (BYOD) model, many organizations also have lost control of corporate endpoints, weakening the traditional enterprise perimeter. Security must now go where the workload goes as it moves through the data center and cloud.

While the attack chain—break in, latch on, expand, gather, exfiltrate—remains the same, the ingenuity of attackers springs eternal. Frequent headlines about catastrophic breaches reflect that the big picture on security continues to shift:

- Attackers have realized that cybercrime does pay, resulting in a rise of advanced persistent threats and other, rapidly mutating malware.
- Nation states have become more sophisticated in their cyberwarfare capabilities. In many nation states, attackers have used state resources to develop sophisticated tools with which they moonlight to make much more money than they do in their day jobs.

The fundamental challenges of securing a cloud platform can certainly keep a security officer up at night. A CISO's goal is to define the organization's security framework and requirements to minimize risk and satisfy compliance with regulations. Implementations have to be auditable.

These requirements may put the CISO at odds with the AppDev executive, who needs a security solution that provides automation wherever possible and flexibly integrates into DevOps processes and pipelines (if it can't be made fully invisible).

How can a cloud platform efficiently and effectively meet the important but contrasting needs of key stakeholders?

Creating a chain of trust

The solution is to create a hardware-rooted chain of trust that verifies the integrity of every relevant component in the cloud platform. **A true chain of trust would start in the host chip firmware and build up through the container engine and orchestration system, securing all critical data and workloads during an application's lifecycle.** The result would be a highly automated, trusted container system.

Hardware is the ideal foundation because it is rooted in silicon, making it difficult for hackers to alter. The chain of trust would be built from this root using the measure-and-verify security model, with each component measuring, verifying and launching the next level. This process would extend to the container engine, creating a trust boundary, with measurements stored in a Trusted Platform Module (TPM) on the host. Attestation software on a different server would verify current measurements against known good values. The container orchestrator would communicate with the attestation server to verify the integrity of worker hosts and any container images deployed on them.

Key takeaway

Make sure the cloud platform supports a policy-managed trust boundary, which is vital to automating security.

Figure 1 represents a chain of trust, rooted in the hardware, that would be involved when adding a new worker to a Kubernetes cluster. Numbers in the illustration correspond to steps 1 through 6 described here. Keep in mind that verifying a measurement on the booting host requires comparison to a known good stored on a separate attestation server.

1. On the worker host, TPM hardware authenticates the system firmware, measuring and verifying the BIOS, including the optional ROMs. It then launches the BIOS.
2. The BIOS measures, verifies and launches the operating system (OS).
3. The OS measures, verifies and launches the Docker container runtime, Docker plugins and all key components that are part of a trusted computing base (TCB).
4. With a Cloud Integrity Technology (CIT) plugin, the Kubernetes master verifies the worker host through the attestation server. Attestation could also involve checking geo-location/boundary information for the Kubernetes cluster, such as blocking use of a worker who is not appropriately geo-located.
5. The Kubernetes master configures the validly attested host as part of the existing cluster, which includes assigning containers to it.
6. The Docker engine on the worker host, communicating through an encrypted connection with the attestation server, would verify the integrity of container images and check them against security policy.

Because it is driven by security policies, the entire container platform automatically and only runs hosts and containers in known good states.

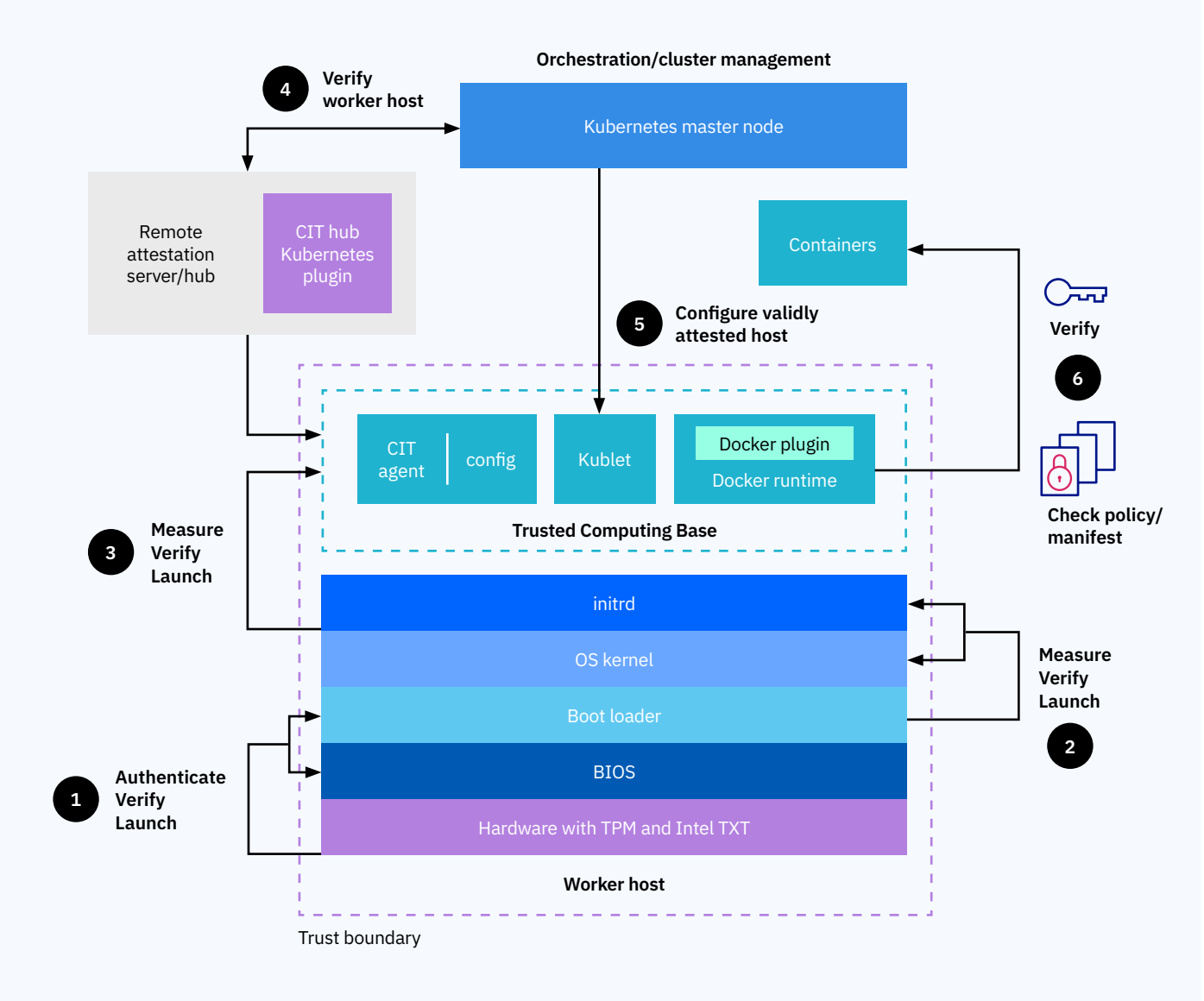


Figure 1. A reference architecture enabling a chain of trust for containers that employs the measure-and-verify security model as the foundation that extends to the Kubernetes orchestration level. See page 3 for full descriptions of the six steps.

Enabling trusted containers

Container systems like Docker have built-in methodologies for creating micro-perimeters around separate elements of the system that help safeguard communication between them.

To evaluate whether containerized apps will be sufficiently protected, ask the cloud platform vendor about these aspects of their Docker implementation:

Are software images kept in a private registry? Cloud platforms based on Docker Registry V2 can assign each organization a secured private image registry where images are stored and shared only by specified users and groups. Adding images to the private registry involves authorizing users to create or copy local images or to import an image directly from a public repository like Docker Hub.

Does the Docker implementation enable encryption of its images?
Encryption protects against tampering with the images in the registry.

Are the Docker daemons that run on the compute hosts set up without direct user access? Are they configured only by the service provider?
The answer to both should be yes. Direct access to the host by other customers could compromise the security of your containers.

Are all Docker daemon sockets secured with transport layer security (TLS) certificates?
TLS combines the advantages of public-key cryptography, external third-party validation and per-session encryption.

Are any privileged Docker containers allowed?
Disallowing privileged containers ensures that containers of other service provider customers do not have access to hard disks on the compute host that may contain your data and applications.

Moving from node trust boundary to trusted cloud

Since the compute node becomes the focus in establishing the chain of trust, and because each node has its own trust boundary, all members of a Kubernetes cluster and pod begin securing the overall workload before any computation and transfer of data occur.

Expect cloud providers to describe and demonstrate their trust technologies. For example, Intel Trusted Execution Technology (Intel TXT), any TPM that complies with specifications 1.2 or 2.0, and Intel CIT are established technologies a provider might use to build a trusted cloud.

- **Intel TXT** defends against software-based attacks aimed at stealing sensitive information by corrupting system or BIOS code, or by modifying the platform's configuration.
- **TPM** is a hardware-based security device that stores the measurements used in the measure-and-verify security process. It helps ensure the system is tamper-free before releasing system control to the next level of software.
- **Intel CIT** builds on the root of trust to provide policy-driven attestation information so that workloads run on verified hardware with compliance in both public and private cloud environments.

Remote attestation is an important step in the trust process, extending beyond the host trust boundary to the container orchestration level. An orchestrator like Kubernetes must be able to verify the integrity of a compute node before deploying containers to it.

To provide remote attestation, cloud vendors may use CIT technologies, which add a further verification step whenever a cloud compute node is assigned to the container environment. Intel CIT, for example, works hand-in-hand with Intel TXT to help ensure the node is still tamper-free and trusted before it is accepted into a container cluster. Intel CIT also provides an extension that makes it easy for a DevOps team to enable security policies for workloads without requiring the application developer to deal with the policies.

Key takeaway

Extending the node-level trust boundary requires a proven encryption solution.

Safeguarding security through separation of resources

The Kubernetes orchestrator also helps secure a cluster by allowing separation of service-provider-managed resources from private elements in an organization's account (Figure 2):

- The Kubernetes dedicated master node and the private image registry with controlled image access can run in the managed network.
- The Kubernetes worker nodes, with containerized workload pods, can be deployed in the organization's infrastructure account on dedicated networks controlled by the organization, not by the provider.

This approach gives DevOps teams a high level of control and provides the isolation CISOs want. Communication between the master and worker nodes would take place through an encrypted network connection, with Kubernetes providing the encryption and keys; an ingress controller would auto-generate TLS certificates for accessing the Kubernetes pods. Using Kubernetes role-based access controls, organizations can set fine-grained restrictions over the resources within clusters.

Policy-driven automation

Kubernetes allows DevOps teams to break system functions into very small atomic elements, each of which can be tied into the base trusted architecture to help ensure that every element permits access and communication according to policy. As teams build out complex microservices architectures, policy-driven automation can control access and routing, making it easy to expand and contract scale for individual applications and their components.

Calico and Istio are two important components of the Kubernetes ecosystem that help with application and workload security. Calico simplifies management of IP addresses assigned to the workloads in a compute node, and programs access control lists in each compute node to enforce security policies. Through policy definitions set up and enforced via labels, Istio provides certificate-based control of communication among microservices within a Kubernetes pod or cluster.

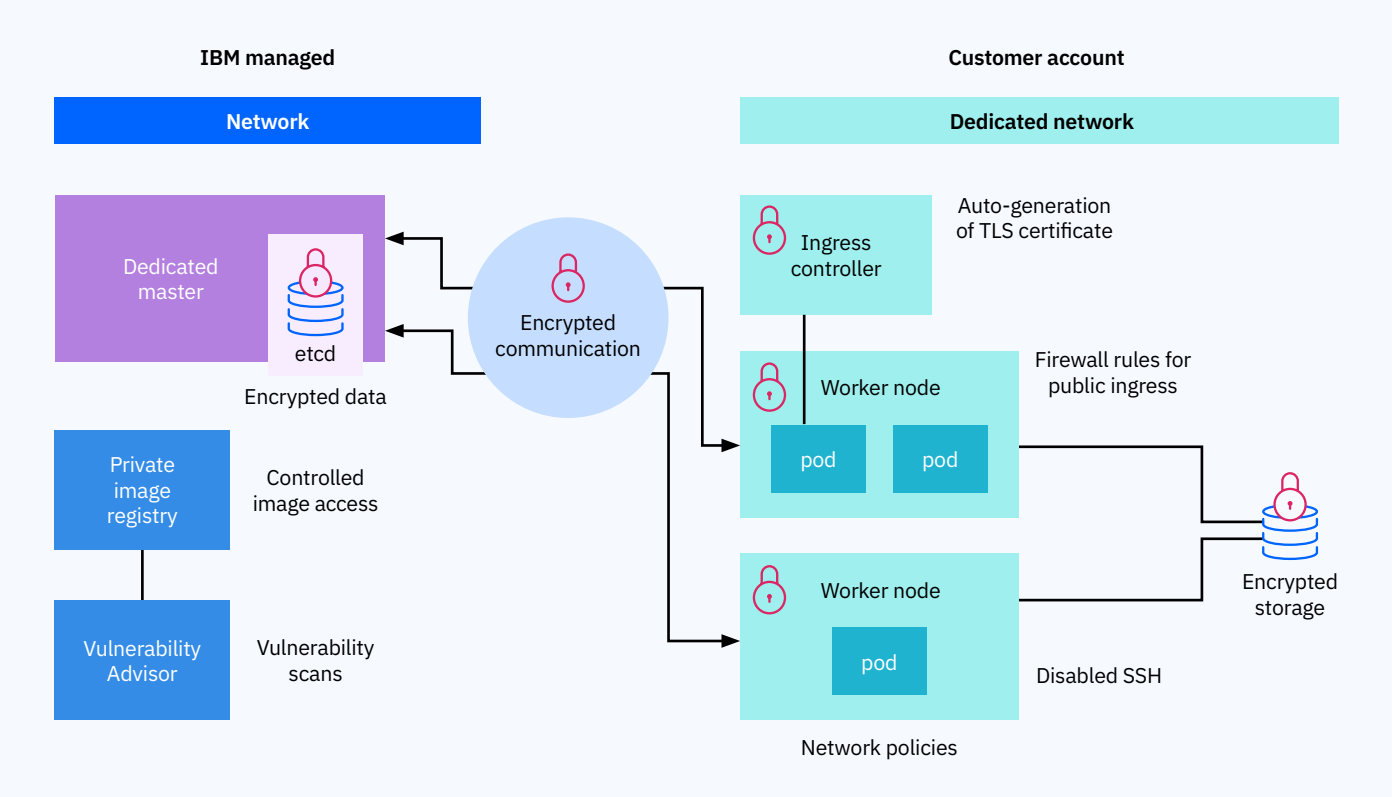


Figure 2. Separation of provider-managed and customer-managed cluster elements.

Extending the benefits of a chain of trust

A fully implemented chain of trust with remote attestation and encryption tied to security policies enables these important capabilities for managing containers, applications and workloads:

- **Transparency and scalability:** With automation made possible through the chain of trust, DevOps teams are freed to work at unimpeded velocity. They need to manage only the security policies against which the trusted container system evaluates its measurements. With the appropriate configuration in place, orchestration more or less automatically scales application resources based on real-time traffic.
- **Geographical workload policy verification:** Smart container orchestration limits movement to only approved locations.
- **Container integrity assurance:** When containers are moved, they are checked to ensure no tampering has occurred during the process. The moved container is verified to be the same as the originally created container.
- **Security for sensitive data:** Encrypted containers can be decrypted only on approved servers in specific locations.
- **Simplified compliance controls and reporting:** A metadata audit trail provides visibility and auditable evidence that critical container workloads run on trusted servers.

Key takeaway

As your team evaluates cloud platforms, ask vendors to explain how trust is established and maintained for the footprint of technology that will host the applications. This is the foundation upon which your organization's business depends to engage customers and hold important data.

Case in point: Alleviating GDPR worries

Suppose you have customers in Europe, and you're concerned with potentially huge liabilities coming into play as the European Union General Data Protection Regulation (GDPR) goes into effect. Because sovereignty requirements and other regulations mean certain kinds of data can't leave the country in which they originated, you need:

- Strong assurance that when you want your workload in a particular locality, it can't and won't go anywhere else.
- Encryption keys for your workload that are managed such that the data can't be decrypted anywhere except where you placed your workload.

Once you have established a hardware-rooted chain of trust, you can tie in all of the elements that are critical to maintaining integrity, managing your keys and guaranteeing locality of your workloads. And you can drive this trust through policies, scaling security along with application deployments.

Scanning static and live containers

Getting started with Docker containers is easy: Developers can pull down any container image publicly available on Docker Hub, for example, avoiding or significantly reducing the time necessary to prepare parts of an image stack. The problem is not knowing for sure what is in that image before deploying it. A necessary practice, therefore, is to scan every image before releasing it into the DevOps pipeline proper. Cloud platforms must provide an efficient way of doing this.

IBM® Cloud Container Service, for example, offers a Vulnerability Advisor (VA) system to provide both static and live container scanning (Figure 3). VA inspects every layer of every image in a cloud customer's private registry to help detect vulnerabilities or malware before image deployment. However, because simply scanning registry images can miss problems such as drift from static image to deployed containers, VA also scans running containers for anomalies. In addition, it provides recommendations in the form of tiered alerts.

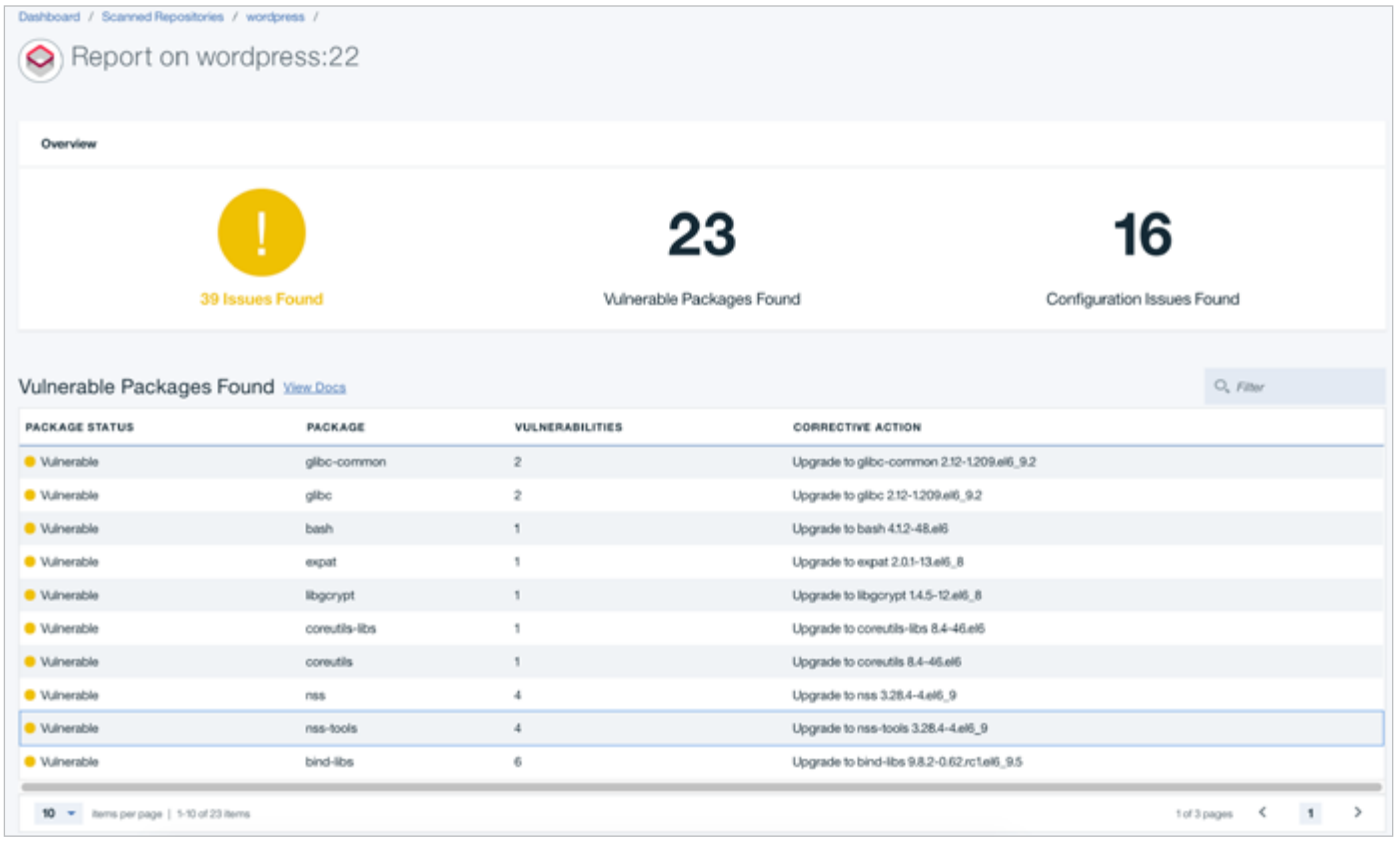


Figure 3. VA integrates with X-Force to rate vulnerabilities based on attack vector, complexity and availability of a known fix.

Cloud isolation technologies

Implied by chip-based technologies, implementing a chain of trust requires the ability to deploy on dedicated hosts that support VPN access. All containers should run as independent, isolated processes on a compute host and should have restricted access to its resources.

Using optimized compute host kernels, a cloud provider should be able to automatically limit the total number of threads and processes that are run on any one compute host. This optimization benefits you by ensuring that the host is not overloaded, which could affect your application performance.

A service provider should also continuously monitor compute hosts to control and remediate fork bombs and other process-level DoS attacks. Security controls governing access to folders, files, network domains and permissions to create and change data should start at the Linux kernel level.

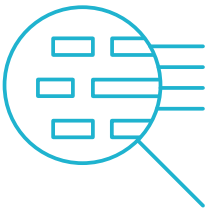
Visibility into cloud security

Operations engineers are used to scrutinizing their on-premises resources, and justifiably expect the same insight into their cloud-based, containerized workloads. To provide this visibility, cloud vendors should automatically log all user and administrative access, whether by the organization or the vendor. A built-in cloud activity tracker can create a trail of all access to the platform and services, giving client organizations access to relevant logs.

Make sure you have the option of integrating all logs and events into your on-premises security operations center (SOC) and security information and event management (SIEM) system. Some cloud service providers offer additional services such as security monitoring with incident management and reporting, real-time analysis of security alerts and an integrated view across hybrid deployments.

IBM QRadar®, for example, is a comprehensive SIEM solution providing a set of security intelligence capabilities that can grow with an organization's needs. It contains machine learning capabilities that train on threat patterns in a way that builds up an intelligent security immune system.

Exploring Vulnerability Advisor



Features of IBM Vulnerability Advisor include the following:

- **Policy violation settings:** With VA, administrators can set image deployment policies based on three types of image failure situations: installed packages with known vulnerabilities, remote logins enabled and remote logins enabled with some users who have easily guessed passwords.
- **Best practices:** VA currently checks 26 rules based on ISO 27000. Checks include settings such as password minimum age, minimum password length and remote logins enabled.
- **Security misconfiguration detection:** VA flags each misconfiguration issue, providing a description of it and recommending a course of action to remediate it.
- **Integration with IBM X-Force®:** VA pulls in security intelligence from five third-party sources and use criteria such as attack vector, complexity and availability of a known fix to rate each vulnerability. The rating system (critical, high, moderate or low) helps administrators quickly understand the severity of vulnerabilities and prioritize remediation.

End-to-end security in service of business needs

Container technology serves application development teams by streamlining and increasing the velocity of their collaborative work in cloud environments. But to deliver those benefits, a cloud platform must meet CISO security requirements without introducing undue friction. Therefore, to meet business goals, DevOps teams need to implement CISO policies through automated security.

A hardware-rooted chain of trust is an effective foundation for this goal. It should include technologies for ensuring trusted containers and enforcing security policies that govern container deployment. The chain-of-trust architecture is designed to meet the urgent need for both security and rapid innovation:

- Security officers can formulate security policies that are automatically applied to every container that is created or moved.
- Each step in the sequence is automated, enabling DevOps teams to quickly build and deploy applications without stopping to add security components.

This architecture protects data and applications from the hardware level to the container orchestration layer of cloud platforms, helping organizations meet compliance regimes such as the EU GDPR, US Federal Risk and Authorization Management Program (FedRAMP) and US Health Insurance Portability and Accountability Act (HIPAA). Organizations define exactly the policies required for their industry and guarantee the elements of assurance.

The IBM point of view

Innovating on the chain of trust is a key focus for IBM and its partners. IBM and Intel have a long partnership dedicated to developing chain-of-trust security solutions, and are now applying their expertise to container-based offerings. The goal: Help organizations deploy containers in a secure yet agile way that enables the development flexibility and cutting-edge microservices architectures today's innovators demand and deserve.

IBM Cloud empowers teams with readily available open source tools for automating deployment and management. And if customers want to deploy workloads on multiple clouds, the cloud platform must enable them to use the same tools consistently across their multi-cloud environment. The future of container security is open, agile, automated wherever possible, and both strongly and intelligently defensive.

The future of container security is open, agile, automated wherever possible, and both strongly and intelligently defensive.

For more information

To learn more about building a chain of trust for container security, visit ibm.com/cloud/container-service

Interested in Security + DevOps? Join our Slack channel and compare notes with the developers on the IBM Cloud Container Service product team.

Stay connected

IBM Cloud Container Service
IBM Cloud Blog

Follow us

@IBMcloud
Facebook

Connect with us

LinkedIn
YouTube

IBM Corporation
1 New Orchard Road
Armonk, NY 10504-1722

Produced in the United States of America, February 2018