EMA™ IT & DATA MANAGEMENT RESEARCH, INDUSTRY ANALYSIS & CONSULTING

Taking a Holistic Approach to Securing Cloud-Native Application Development

Declarative Security for Your Modern App Across Clouds

May 2023 EMA eBook
Prepared for Cisco

By **Torsten Volk**, Managing Research Director
Hybrid Cloud, Software-Defined Infrastructure, and Machine Learning

Table of Contents

Cloud-Native Security is Business-Critical

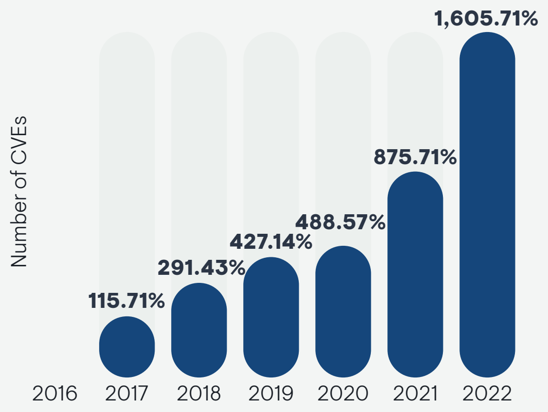
Cloud-native application security has become the primary challenge in app modernization, hindering developer productivity, efficient operations, and optimal risk management. This prevents organizations from delivering better products faster, cheaper, and more often than the competition. Developers spend much of their time implementing and following corporate security policies while introducing more tools, open source libraries, container images, and data sources into the application stack. Operators struggle to keep systems secure without having sufficient visibility into the application workloads running on these systems, and security personas struggle to create, monitor, and enforce compliance rules without significantly slowing down developers. This situation frequently leads to tradeoffs between security, resiliency, and developer productivity that can either land the entire organization in news headlines or dramatically slow down the ability to deliver competitive products.

In response to these security challenges, in May of 2022 the US government acknowledged the importance of focusing on application security by issuing an executive order (Executive Order 14028) Its intent is to make software vendors prove the compliance of their supply chain with security best practices to minimize the risk of vulnerabilities introduced through third-party code and cloud services.

Securing the Software Supply Chain is Key

To prove the security and resiliency of their applications, software vendors must keep a complete record of all third-party components introduced into their supply chain, when the components were introduced, and what measures were taken to ensure that the software and all of its dependencies were developed in a secure manner. While these core security requirements all make sense, most enterprises struggle with their implementation. Fundamentally, this is due to the old way of doing things no longer working in today's world of dynamic complexity. Before, applications often consisted of numerous distributed microservices developed by separate teams, based on multiple different runtimes, driven by different databases and different types of object storage, with the ability to quickly move across Kubernetes clusters, serverless platforms, and different clouds, and released multiple times per day.

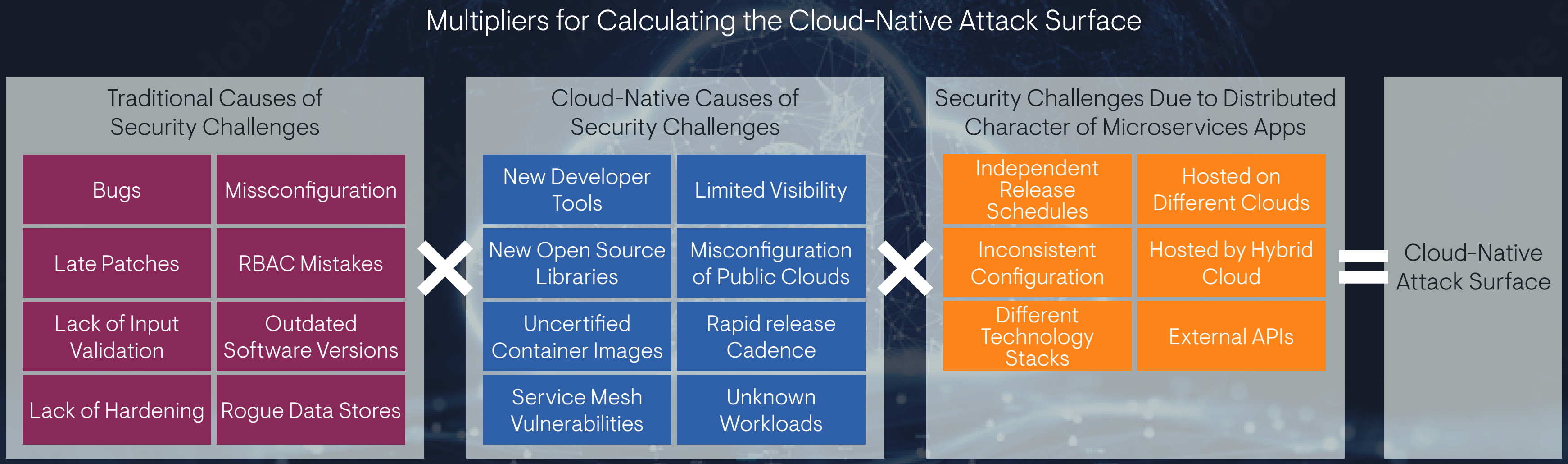
The number of cloud-native vulnerabilities doubled over the previous 12 months



The chart is based on the official catalog of Common Vulnerabilities and Exposures (CVE) maintained by the MITRE Corporation.

A New Equation for Cloud-Native Security

Organizations are faced with a new equation that includes numerous multipliers of application complexity. Traditional security challenges (1) are amplified by challenges that are specific to cloud-native applications (2). We need to factor in the distributed character of microservices applications with separate development teams making different technology choices and following a different release schedule (3). These challenges require a holistic and application-centric approach toward cloud-native security, protecting against vulnerabilities across runtimes, containers, and Kubernetes clusters. Finally, organizations need to protect their software supply chain against vulnerabilities of the actual application code, independently of whether this code is hosted on a self-managed Kubernetes cluster, a cloud-managed cluster, or a serverless environment.


Multipliers for Calculating the Cloud-Native Attack Surface
Traditional Causes of Security Challenges
Bugs
Missconfiguration
Late Patches
RBAC Mistakes
Lack of Input Validation
Outdated Software Versions
Lack of Hardening
Rogue Data Stores
Cloud-Native Causes of Security Challenges
New Developer Tools
Limited Visibility
New Open Source Libraries
Misconfiguration of Public Clouds
Uncertified Container Images
Rapid release Cadence
Service Mesh Vulnerabilities
Unknown Workloads
Security Challenges Due to Distributed Character of Microservices Apps
Independent Release Schedules
Hosted on Different Clouds
Inconsistent Configuration
Hosted by Hybrid Cloud
Different Technology Stacks
External APIs
Cloud-Native Attack Surface

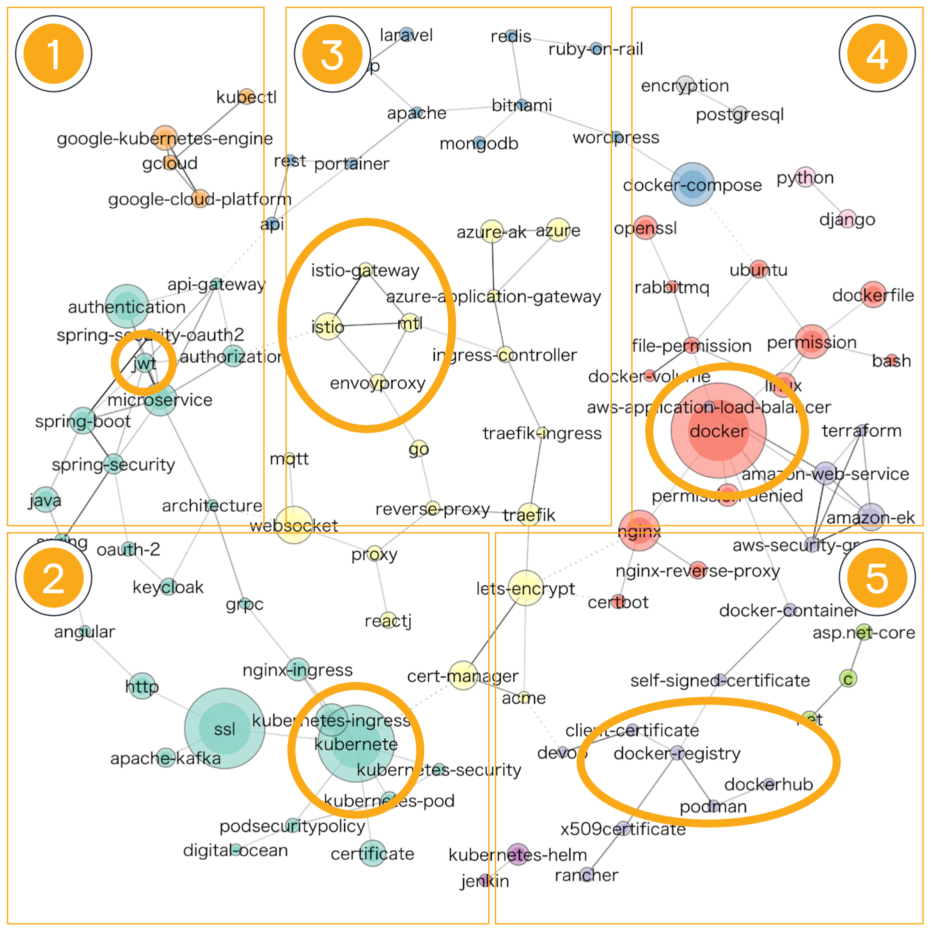
Cloud-native application architectures add two complexity multipliers (2 and 3) to the traditional causes of cybersecurity challenges (1). This results in a drastic increase in the size of the cloud-native attack surface.

Top Five Cloud-Native Security Areas for Developers

The chart shows the relationships between the most common security topics related to cloud-native applications experienced by developers on a daily basis (based on questions asked in the StackOverflow support forum). We can divide this chart into five problem clusters (framed in orange):

1. Microservices and APIs
2. Kubernetes
3. Service Mesh
4. Container Runtime
5. Container Registry

1. **Microservices and APIs**
   The fact that microservices communicate via API connections makes them more susceptible to attacks than traditional applications that communicate directly via internal function calls. Scanning the network for unprotected or over-privileged APIs is a simple and therefore popular approach for attackers to gain access to enterprise systems. This is similar to a car thief checking the doors of rows of cars to find the ones that are unlocked. Then, the thief can take all valuables inside of the car, but he or she may also check in the usual places to see if the owner may have left the key and maybe even the remote control to their garage. This shows the importance of optimal authentication and authorization, not only to secure one microservice, but to prevent a successful attack on this microservice from enabling the attacker to access further microservices or even the underlying Kubernetes cluster or database. The chart shows authentication and authorization, and JWT (JSON Web Token) as critical problem areas that are directly connected to the API gateway. The API gateway is key for microservices security since misconfigurations can provide excessive access privileges to external users, enabling them to penetrate sensitive data, execute malicious code, and use their inflated access level to jump over to other microservices, either directly or via the underlying Kubernetes host.



2. **Kubernetes**
   As the universal platform for running cloud-native applications, Kubernetes presents a vast attack surface that often stretches across applications, data centers, and clouds. Finding one entry point, such as an unsecured kubelet (Kubernetes agent), an unprotected API, or an infected container image, can open the door for a vast variety of attacks across application clusters. Due to the size and the open character of the Kubernetes universe, in addition to taking advantage of Kubernetes vulnerabilities directly, attackers can find their way into the corporate network through a variety of applications with direct Kubernetes integration. The Linux operating system, the Argo DevOps pipeline automation platform, Docker, and the GitHub source code management platform all are potentially viable attack vectors. Once inside, the malware searches for entry points to further expand its access across Kubernetes clusters within the organization.

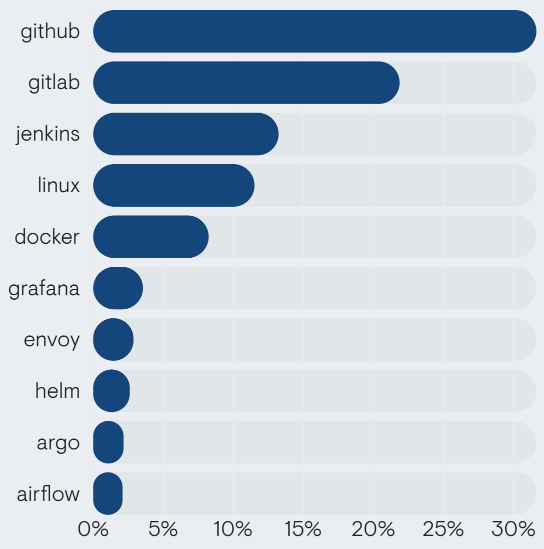
   Therefore, we need to regard ingress control and secure socket layer (SSL) certificates as the key topics (see chart) when it comes to defending Kubernetes clusters.

3. **Service Mesh**
   The service mesh controls the communication between microservices and can therefore become an impactful point of access for attackers. Injecting malicious sidecar containers into application containers can allow attackers to reroute traffic and use the container's access credentials to access other microservices. This makes tight ingress control critical because the ingress controller can protect applications from outside access and from access by infected containers within the cluster.

4. **Container Registry**
   Embedding malware into container images on unauthorized registries can be an effective point of entry for attackers. Therefore, implementing strict certification rules, requiring identification, and enforcing immutability are critical.

5. **Container Runtime**
   Protecting the container runtime includes securing the Linux operating system, ensuring secure file access, scanning resources generated based on infrastructure-as-code blueprints, ensuring encryption, and securely configuring the ingress controller.

Top 10 Kubernetes-Related Software Products by Number of Security Vulnerabilities



The chart compares the number of security vulnerabilities of software products that are typically part of a Kubernetes stack identified in the CVE catalog for 2022.

The following examples of cloud-native vulnerabilities introduced by some of today's most popular software projects and products demonstrate the importance of a holistic approach toward cybersecurity in order to proactively detect and remediate these issues in near-real time.

High-Severity Vulnerabilities by DevOps Toolsets

Grafana contains an authentication bypass vulnerability that allows authenticated and unauthenticated users to view and delete all snapshot data, potentially resulting in complete snapshot data loss.
(CVE-2021-39226)

Helm is a tool for managing Charts, pre-configured Kubernetes resources. Versions prior to 3.10.3 are subject to Uncontrolled Resource Consumption, resulting in Denial of Service...this issue has been patched in 3.10.3. SDK users can validate strings supplied by users that won't create large arrays causing significant memory usage before passing them to the _strvals_ functions.
(CVE-2022-23524)

Atlassian: Multiple API endpoints of Atlassian Bitbucket Server and Data Center contain a command injection vulnerability where an attacker with access to a public Bitbucket repository, or with read permissions to a private one, can execute code by sending a malicious HTTP request.
(CVE-2022-36804)

Capsule is a multi-tenancy and policy-based framework for Kubernetes...start privileged containers that would be able to create a generic Kubernetes privilege escalation. Patches have been released for version 0.1.3. No known workarounds are available.
(CVE-2022-46167)

IBM Cloud Kubernetes Service is affected by security vulnerabilities in the Kubernetes API server that may allow access to secure endpoints in the control plane network (CVE-2022-3294). Service allows users authorized to list or watch one type of namespaced custom resource cluster-wide to read custom resources of a different type in the same API group without authorization.
(CVE-2022-3162)

Azure Cosmos DB: The flaws created a loophole, giving users the ability to access other databases that weren't theirs. A range of organizations was impacted by the issue, including several Fortune 500 companies. (Reuters.com, August, 2021)

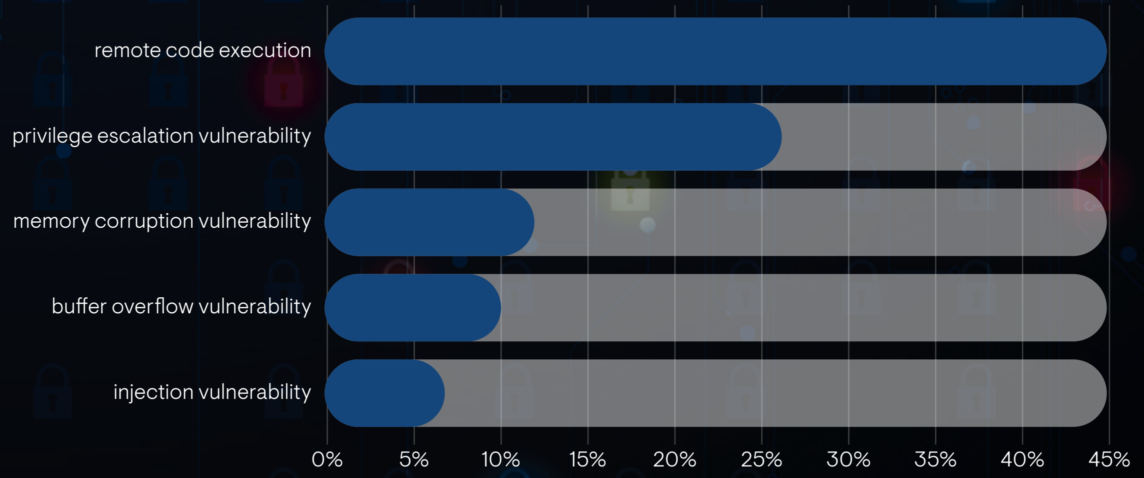
Flux2 is a tool for keeping Kubernetes clusters in sync with sources of configuration, and Flux's Helm controller is a Kubernetes operator that allows one to declaratively manage Helm chart releases. In a shared cluster multi-tenancy environment, a tenant could create a Helm release that makes the controller panic, denying all other tenants from their Helm releases being reconciled. Patches are available in flux2 v0.32.0 and Helm-controller v0.23.0.
(CVE-2022-36049)

containerd is an open source container runtime. A bug was found in containerd's CRI implementation where a user can exhaust memory on the host. If the user's process fails to launch due to, for example, a faulty command, the go-routine will be stuck waiting to send without a receiver, resulting in a memory leak.
(CVE-2022-31030)

The Case for a Holistic Approach to Security

Remote code execution, privilege escalation, and memory corruption are the most frequently occurring types of exploited security vulnerabilities in the CISA catalog (see chart). All it takes is one developer or DevOps engineer adding additional permissions to a user role or applying additional user roles to an account to unintentionally make a service expose any number of code objects and provide unintended remote access to confidential parts of the application, its data sources, or even to the entire Kubernetes cluster.

Top Five Exploited Vulnerabilities Based on CISA



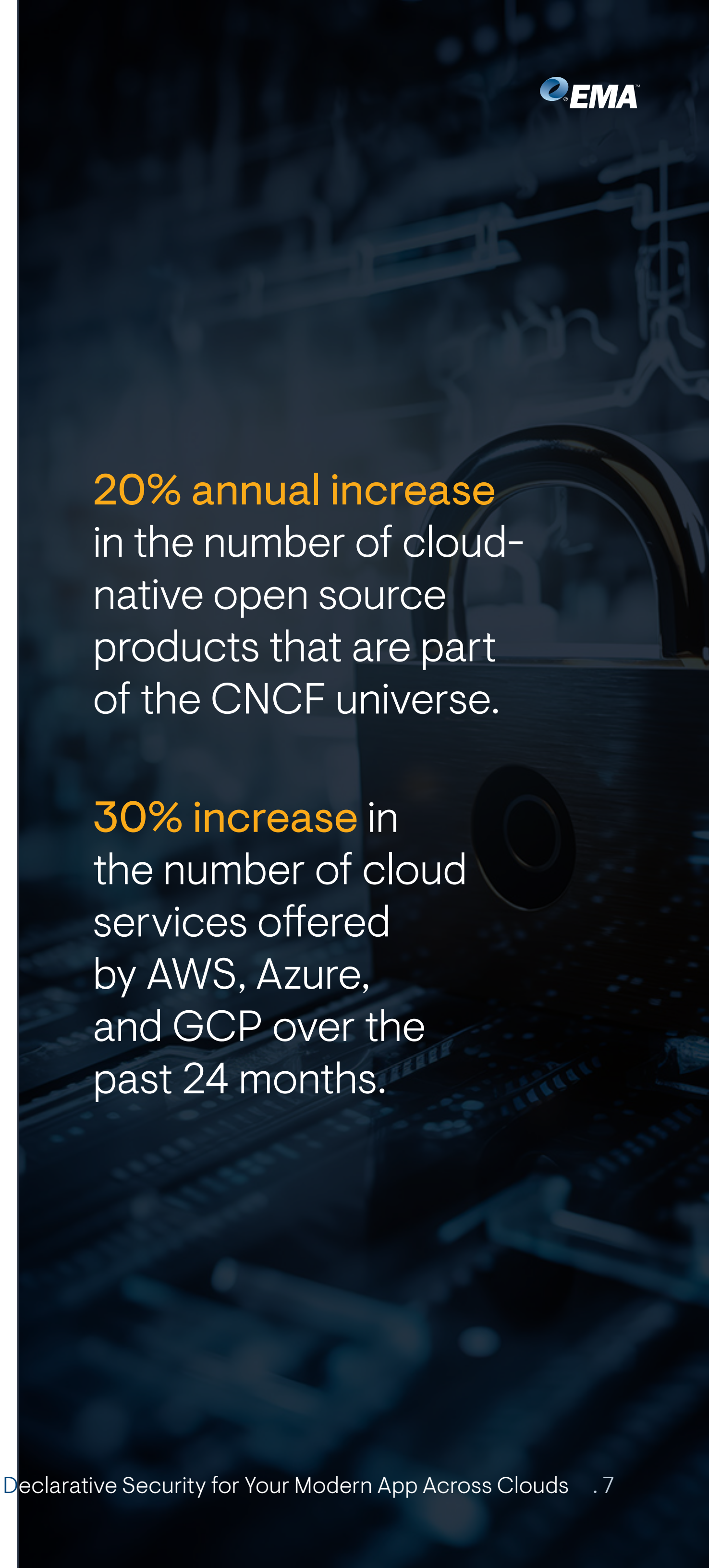
These are the top five types of vulnerabilities recorded by the US Cybersecurity and Infrastructure Security Agency (CISA) in 2022.

Source: CISA

These findings go directly in line with the previously identified five critical cloud-native security topics that illustrate how quickly and comprehensively attackers can penetrate and jump between applications consisting of partially shared and loosely coupled microservices. This challenge is fueled by five factors that illustrate the continued expansion of the cloud-native universe in terms of complexity and dynamic change.

- **Public Cloud Services:** The number of cloud-native technologies for developers to consume from AWS, Azure, or GCP increased by 30% over the previous 24 months. Additionally, we can observe 100-150 monthly changes to existing cloud APIs.
- **Open Source Products:** The CNCF number of open source products and projects that are part of the CNCF universe increases by 20% annually.
- **Release Frequency:** Developers are expected to respond to changing customer requirements within an increasingly short period. Instead of one or two annual releases, we often observe weekly or even daily release cycles.
- **Serverless:** Serverless functions are the fastest growing offerings by AWS, Azure, and GCP. This requires organizations to secure application code independently from its underlying infrastructure.
- **Machine Learning:** The rising importance of enabling developers to leverage machine learning or AI-driven prediction models to enhance the user experience and overall value of their applications requires organizations to provide access to the required data sources that are needed to drive these models.

These challenges, in combination with the fact that most organizations are still in the early stages of application modernization and digital transformation as a whole, illustrate that merely shifting left compliance and security measures is not enough. We need to start left.



20% annual increase in the number of cloud-native open source products that are part of the CNCF universe.

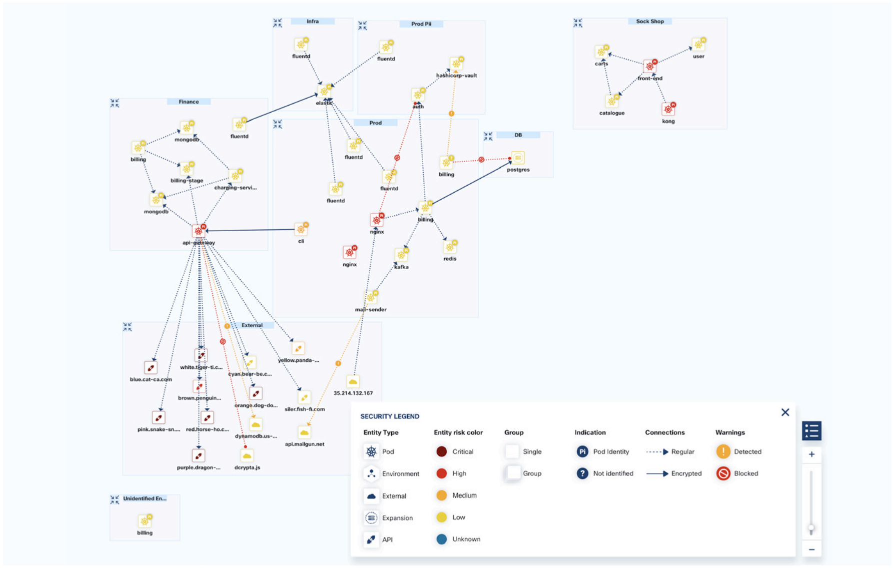
30% increase in the number of cloud services offered by AWS, Azure, and GCP over the past 24 months.

Taming Cloud-Native Complexity With Ten Critical Requirements

Successfully taming cloud-native complexity in order to ensure continuous security requires a complete paradigm shift toward the policy-driven and therefore declarative definition, implementation, monitoring, enforcement, and remediation of application security. Security personas can proactively define policies for a policy engine to translate into specific sets of rules to consistently monitor, enforce, and revise across clusters. Here, it is crucial to embed these policies from the very start into the GitOps pipeline controlled by current infrastructure-as-code (IaC) tools, such as Terraform, Ansible, Amazon CloudFormation, Helm, or PowerShell. Policy-driven app security enables security teams, cloud engineers, platform teams, and application operations teams to collaboratively and continuously define and manage guardrails that ensure the implementation of a secure software supply chain without slowing down developers.

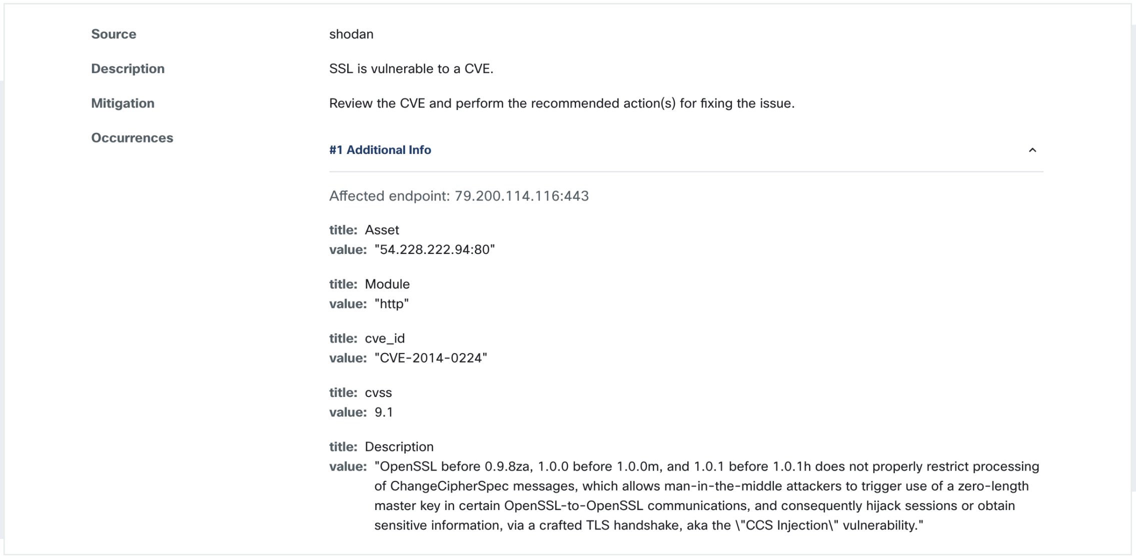
1. **Ensure Continuous Real-Time and Application-Centric Visibility**
   Real-time and historic visibility of the end-to-end application stack, including the Linux operating system, the Kubernetes cluster, container and serverless runtimes, and application workloads, is essential to understand observability data within its context. Showing all relevant roles, policies, secrets, data sources, and any other elements related to each one of those entities allows DevOps to gain an instant overview of anything that may impact application security. Due to the dynamic, loosely-coupled character of Kubernetes apps, it is important to ensure visibility in a near-real time manner and at a high data resolution in order to pick up potentially meaningful, but very brief, events.

Showcase: Cisco Panoptica - Real-Time Application-Centric Visibility



Cisco Panoptica continuously maps relationships and dataflows between application environments and Kubernetes resources across clusters. We instantly notice the items with the highest level of risk attached in red and the ones with low risk in yellow. The dotted lines show unencrypted connections between entities, while the solid lines show encrypted connections. The connections in red were blocked for risk mitigation, while the orange connections show that there are warnings, but the connection is still active. At the bottom of the chart, we notice a cluster of externally-facing APIs all mapped to internal services by the API gateway. The API gateway itself is connected to its CLI, which shows medium risk. We also notice a production mail server exposed to an external cloud service without encryption in the middle and without going through ingress control.

2. **Continuously Monitor and Enforce Best Practices Across the Stack**
   Continuously monitoring and enforcing best practices across the end-to-end stack reduces operational risk by hardening the application environment, including its workloads and the underlying Kubernetes clusters, against standard exploits, such as scanning for unprotected or over-privileged APIs, Kubernetes components that are needlessly exposed to the internet, or container workloads with direct host access. Considering that most attacks are made possible by simple misconfigurations and by insufficient attention toward patching known software issues, continuously monitoring application stacks throughout their development lifecycle is key. The Center for Internet Security (CIS) provides a set of benchmarks that can be translated into enforceable security and compliance rules to obtain a baseline for the implementation of best practices for app security.
3. **Provide Actionable Insights**
   Instead of flooding DevOps with event streams, alerts, metrics, and dashboards containing isolated data points, we need to contextualize events based on their relationships and interdependencies with each part of the application stack, the underlying infrastructure, and external sources, such as the CIS benchmarks and CVE records. This contextualized information helps provide the insights needed for potential auto-remediation, auto-optimization, or human intervention.
4. **Facilitate Collaboration**
   App developers, cloud engineers, DevOps engineers, and security personas need to be able to collaborate to enable each persona to continuously contribute to optimizing application security. Security engineers define declarative security guardrails (see screenshot) for declarative monitoring and enforcement, while developers consume these guardrails through IaC tools, such as Terraform, CloudFormation, Ansible, Helm, or Powershell and contribute application-specific rules. This enables operations roles to continuously monitor general and application-specific security roles, all in one place, to manage operational risk and ensure audit readiness.



Source	shodan
Description	SSL is vulnerable to a CVE.
Mitigation	Review the CVE and perform the recommended action(s) for fixing the issue.
Occurrences	#1 Additional Info

```
Affected endpoint: 79.200.114.116:443

title:  Asset
value:  "54.228.222.94:80"

title:  Module
value:  "http"

title:  cve_id
value:  "CVE-2014-0224"

title:  cvss
value:  9.1

title:  Description
value:  "OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h does not properly restrict processing
        of ChangeCipherSpec messages, which allows man-in-the-middle attackers to trigger use of a zero-length
        master key in certain OpenSSL-to-OpenSSL communications, and consequently hijack sessions or obtain
        sensitive information, via a crafted TLS handshake, aka the \"CCS Injection\" vulnerability."
```

Showcase: Cisco Panoptica - Actionable Insights

In this example, Panoptica identified a vulnerable version of the SSL communication protocol. The alert identifies the affected asset and recommends the review and remediation based on CVE-2014-0024.

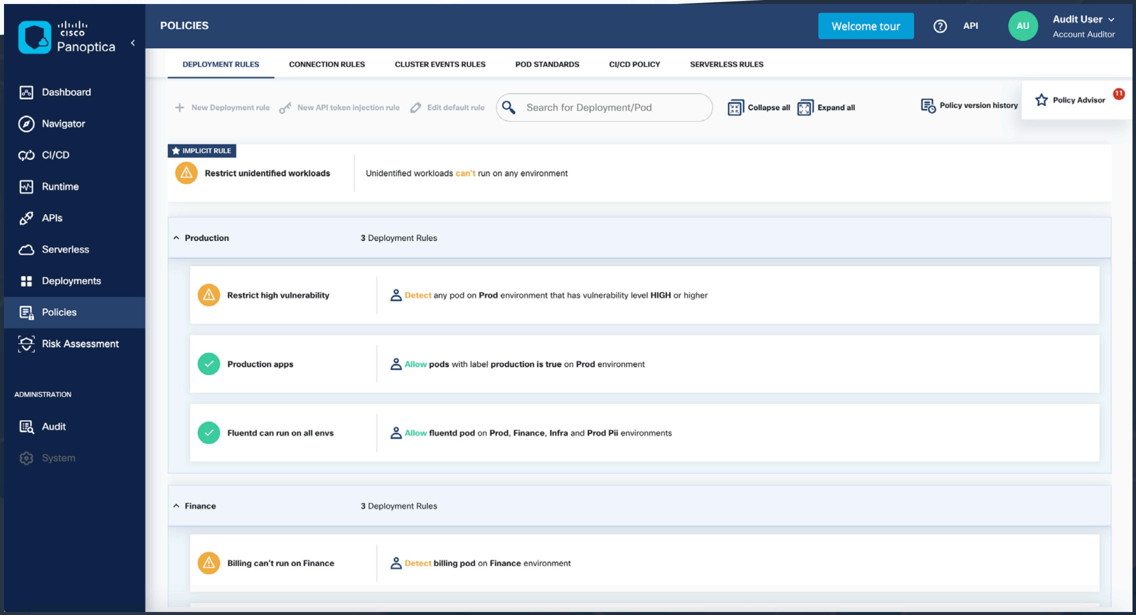
5. **Continuously Scan for Known Vulnerabilities**
All installed software packages, workloads, and infrastructure components should be continuously monitored for known vulnerabilities based on CVE records in order to receive instant warnings of known vulnerabilities. These vulnerabilities can either be addressed through software patches or temporarily remediated through configuration changes and security controls, or simply by adding them to the corporate risk calculation.

6. **Get Continuous and Rapid Visibility Into APIs**
We typically find three different types of APIs in the enterprise. Internal APIs and APIs for paid services, such as payment processing, present a significantly lower risk compared to free API services used ad hoc by app developers to get the job done. Therefore, enterprises require continuous and rapid visibility into all three types of APIs to be able to ensure consistent governance. Identified vulnerabilities need to be reported by priority and impact on the overall application stack and Kubernetes clusters.

7. **Continuously Scan Kubernetes Clusters for Misconfigurations and Vulnerabilities**
Since Kubernetes is tightly intertwined with applications and workloads, and since Kubernetes configuration and container management are complex, continuously discovering and scanning Kubernetes clusters for misconfigurations and software vulnerabilities that could open the door to potential intruders is important. The goal is to minimize the attack surface of Kubernetes itself while at the same time detecting application workloads with the potential of creating additional entry points for attacks.

Showcase: Cisco Panoptica - Declarative Security Guardrails

Panoptica provides a dashboard for the central, declarative definition of compliance guardrails related to app deployment, connections between microservices, cluster events, POD placement and configuration, CI/CD, and serverless functions.



IMPLICIT RULE
Restrict unidentified workloads | Unidentified workloads can't run on any environment

Production 3 Deployment Rules
Restrict high vulnerability | Detect any pod on **Prod** environment that has vulnerability level **HIGH** or higher
Production apps | Allow **pods** with label **production is true** on **Prod** environment
Fluentd can run on all envs | Allow **fluentd pod** on **Prod**, **Finance**, **Infra** and **Prod Pii** environments

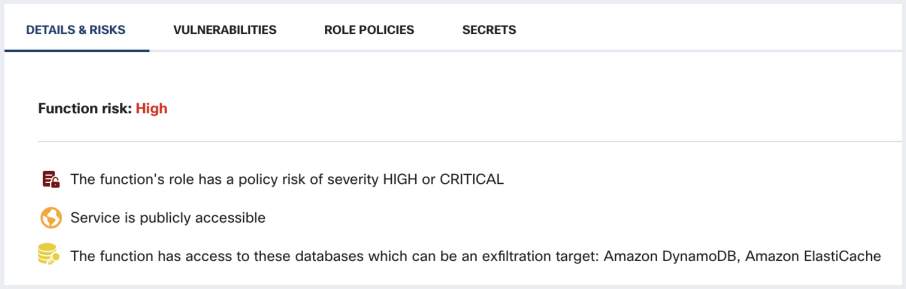
Finance 3 Deployment Rules
Billing can't run on Finance | Detect **billing pod** on **Finance** environment

8. **Enforce Compliance in CI/CD**
   In a world of continuous technology changes, frequent releases, and independent product teams, enforcing compliance starts at the build process by checking for any unauthorized libraries, container images, user roles, data sources, machine learning models, cloud services, Helm charts, and integration points with existing systems. We then need to prevent changes to the configuration or content of container workloads in production.

9. **Mitigate Risk With Trusted Images**
   Scanning for unidentifiable container images and workloads helps prevent containers originating from unknown or blacklisted registries to run on corporate Kubernetes clusters.

10. **Monitor Serverless Functions**
   Serverless functions should be monitored for their ability to provide potential attackers with access to corporate resources: for example, by using over-privileged accounts or by providing direct access to other corporate public cloud services, such as databases, servers, or logs. To optimize the security posture, inactive functions should be flagged for removal as appropriate.

Showcase: Cisco Panoptica - monitoring serverless functions to point out risk to DynamoDB and ElastiCache data stores.


DETAILS & RISKS | VULNERABILITIES | ROLE POLICIES | SECRETS

Function risk: High

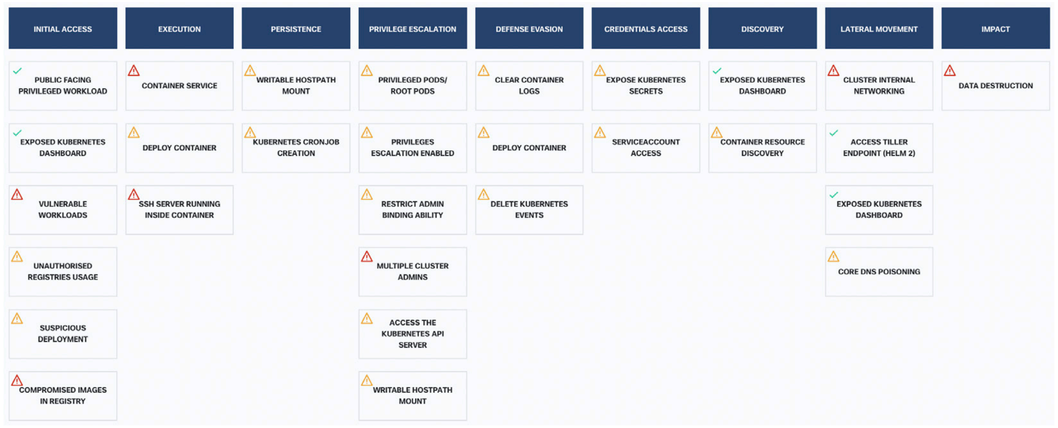
The function's role has a policy risk of severity HIGH or CRITICAL

Service is publicly accessible

The function has access to these databases which can be an exfiltration target: Amazon DynamoDB, Amazon ElastiCache

The screenshot shows an alert that warns of a vulnerability to data theft from DynamoDB and ElastiCache due to the use of an over-privileged user role to run a serverless function.

Showcase: Cisco Panoptica - Kubernetes dashboard showing current Kubernetes-related vulnerabilities in red and orange.



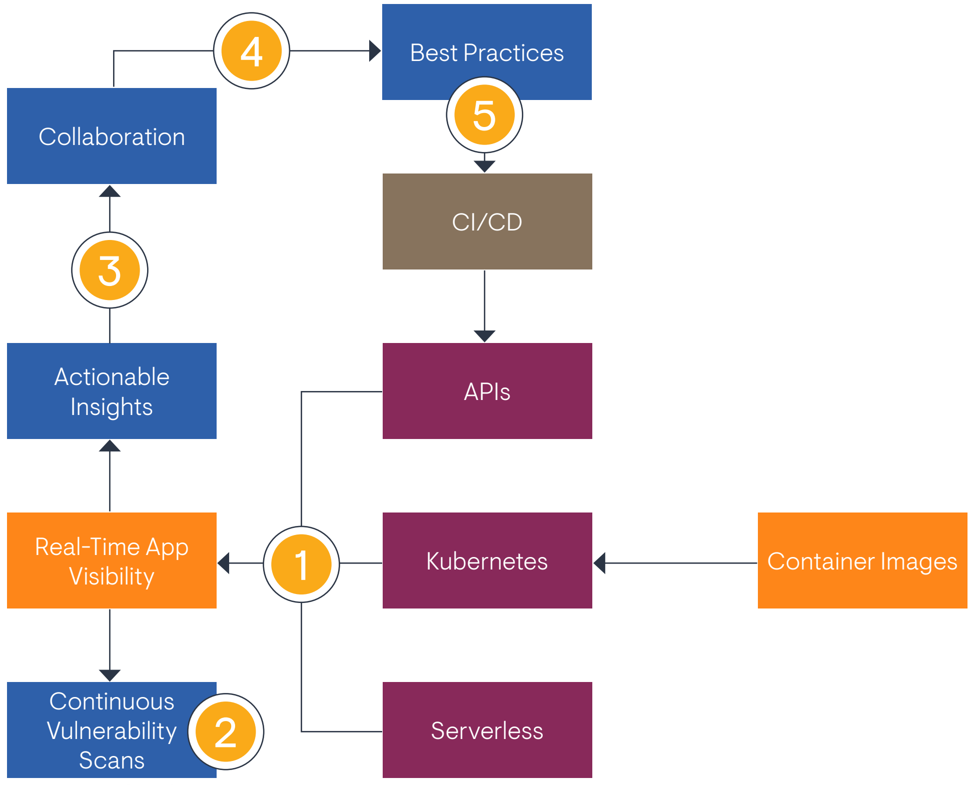
INITIAL ACCESS	EXECUTION	PERSISTENCE	PRIVILEGE ESCALATION	DEFENSE EVASION	CREDENTIALS ACCESS	DISCOVERY	LATERAL MOVEMENT	IMPACT
✓ PUBLIC FACING PRIVILEGED WORKLOAD	⚠ CONTAINER SERVICE	⚠ WRITABLE HOSTPATH MOUNT	⚠ PRIVILEGED PODS/ ROOT PODS	⚠ CLEAR CONTAINER LOGS	⚠ EXPOSE KUBERNETES SECRETS	✓ EXPOSED KUBERNETES DASHBOARD	⚠ CLUSTER INTERNAL NETWORKING	⚠ DATA DESTRUCTION
✓ EXPOSED KUBERNETES DASHBOARD	⚠ DEPLOY CONTAINER	⚠ KUBERNETES CRONJOB CREATION	⚠ PRIVILEGES ESCALATION ENABLED	⚠ DEPLOY CONTAINER	⚠ SERVICEACCOUNT ACCESS	⚠ CONTAINER RESOURCE DISCOVERY	✓ ACCESS TILLER ENDPOINT (HELM 2)	
⚠ VULNERABLE WORKLOADS	⚠ SSH SERVER RUNNING INSIDE CONTAINER		⚠ RESTRICT ADMIN BINDING ABILITY	⚠ DELETE KUBERNETES EVENTS			✓ EXPOSED KUBERNETES DASHBOARD	
⚠ UNAUTHORISED REGISTRIES USAGE			⚠ MULTIPLE CLUSTER ADMINS				⚠ CORE DNS POISONING	
⚠ SUSPICIOUS DEPLOYMENT			⚠ ACCESS THE KUBERNETES API SERVER					
⚠ COMPROMISED IMAGES IN REGISTRY			⚠ WRITABLE HOSTPATH MOUNT					

The screenshot shows a prioritized and categorized list of Kubernetes-related security vulnerabilities. Each item links to a description that provides recommended remediation and resolution actions.

The Five Components of a Holistic Approach Toward Cloud-Native Security

Real-time visibility into serverless functions, Kubernetes clusters, application containers running on these clusters, and application APIs.

1. Continuous vulnerability scans across the entire stack.
2. Actionable insights based on contextualized events.
3. DevOps, platform engineers, cloud engineers, SREs, and security engineers collaboratively revise and refine best practices.
4. Best practices are implemented into each new release. Each release contains declarative policies to automatically enforce compliance. No changes are made outside of the CI/CD pipeline
5. Taking a holistic approach hardens the entire software development lifecycle from the beginning and prevents the introduction of production vulnerabilities through enforcing immutability.



EMA Perspective

Organizations need to embrace the fact that developers continuously seek out new and improved tools for enhanced productivity. Individual development teams need the freedom to autonomously select their preferred development tools and technology stacks. This typically leads to heterogeneous application environments across product teams with loosely coupled microservices communicating via standard APIs. Due to the often distributed and continuously evolving character of microservices applications, it is critical for organizations to automatically keep track of relationships and interdependencies between internal and external microservices and applications. This is the precondition for securing containers, APIs, and serverless functions in an application-centric manner without introducing restrictions that may slow down developers. Instead, IT operations teams need to provide guardrails that protect the organization from bugs, configuration errors, and vulnerabilities of third-party services.

As the service mesh is responsible for establishing secure connections between continuously changing microservices and therefore is always up to date on relationships and interdependencies, it is the ideal place to continuously enforce security and compliance independently of whether the microservice is running on a container in the corporate data center or on a managed Kubernetes cluster in the public cloud. Embedding real-time data from the service mesh into the CI/CD pipeline is the foundation for DevSecOps as developers, operators, and security engineers can observe the impact of new code, configuration changes, and the introduction of new technologies on each new release.

About Enterprise Management Associates, Inc.

Founded in 1996, Enterprise Management Associates (EMA) is a leading industry analyst firm that provides deep insight across the full spectrum of IT and data management technologies. EMA analysts leverage a unique combination of practical experience, insight into industry best practices, and in-depth knowledge of current and planned vendor solutions to help EMA's clients achieve their goals. Learn more about EMA research, analysis, and consulting services for enterprise line of business users, IT professionals, and IT vendors at **www.enterprisemanagement.com**. You can also follow EMA on **Twitter** or **LinkedIn**.

2770 Arapahoe Road, Ste 132 - 544, Lafayette, CO 80026 +1 303.543.9500 **www.enterprisemanagement.com** 4275.042123