cloud security alliance®

Challenges in Securing Application Containers and Microservices

Integrating Application Container Security Considerations into the Engineering of Trustworthy Secure Systems

ABSTRACT

Application containers and a microservices architecture are being used to design, develop, and deploy applications leveraging agile software development approaches such as Development Operations (DevOps). Security must be embedded into these software development approaches. This document identifies challenges in securing application containers and microservices in the engineering of trustworthy secure systems, through the lenses of the Developer, Operator, and Architect.

ACKNOWLEDGEMENTS

Application Containers and Microservices Working Group Co-chairs:

Anil Karmel
Andrew Wild

Lead Authors:

John Kinsella
Cem Gurkok
Frank Geck

Contributors:

Jeff Barnes
Randall Brooks
Ramaswamy Chandramouli
Madhav Chablani
Atul Chaturvedi
Aradhna Chetal
Joshua Cuellar
Joshua Daniel
Shyamkant Dhamke
Michele Drgon
Michael Green
Michaela Iorga
Amir Jerbi
Juanita Koilpillai
Yin Lee
Aaron Lippold
Vishwas Manral
James McCloskey
Ki-Hong Min
Lloyd Osafo
John Osborne
Mark Potter
Alex Rebo
Michael Roza
Ed Santiago
Kina Shah
Shankar
Ken Stavinoha
Shanthi Thomas
David Wayland
Shawn Wells
John Wrobel
Mark Yanalitis
Ashish Kurmi
James Yaple
Vrettos Molous

CSA Staff:

Hillary Baron
Marina Bregu

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Application containers and a microservices architecture, as defined in NIST SP 800-180, are being used to design, develop, and deploy applications leveraging agile software development approaches such as DevOps. The security of application components needs to be considered throughout the software development life cycle (SDLC). NIST 800-160, Systems Security Engineering, defines the need for trustworthy secure systems based on a wide variety of stakeholder needs. This present document identifies challenges in securing application containers and microservices through the lens of the Developer, Operator, and Architect.

Challenges were scored by contributors to this document based on ten weighted questions applied against application container and microservices features. A detailed explanation of challenges from Developer, Operator, and Architect perspectives provides context from these unique viewpoints.

1 INTRODUCTION

1.1 Purpose and Scope

The purpose of this paper is to present the challenges Architects, Developers, and Operators will encounter when designing, deploying, and operating secure application containers and microservices. Secure DevOps necessitates security as Architects, Developers, and Operators work together to engineer trustworthy secure systems.
The scope of this document is limited to application containers and microservices.

1.2 Document Structure

The remainder of this document is organized into the following sections and appendices:

- Section 2 introduces application container and microservices challenges, including the scoring methodology as applied against said challenges.
- Section 3 provides a detailed description of application container challenges through the lens of the Developer, Operator, and Architect.
- Section 4 provides a detailed description of microservices challenges through the lens of the Developer, Operator, and Architect.
- Appendix A provides common acronyms and abbreviations used throughout this document.
- Appendix B provides a glossary of selected terms from this document.
- Appendix C provides a list of references cited in the document.

1.3 Audience

The intended audience for this document is application Developers, application Architects, system and security administrators, security program managers, information system security officers, and others who have responsibilities for or are otherwise interested in the security of application containers and microservices in a software development life cycle (SDLC).

This document assumes that readers have some operating system, networking, and security expertise, as well as expertise with application containers, microservices and agile application development approaches such as DevOps. Because of the constantly changing nature of application container technologies, readers are encouraged to take advantage of other resources, including those listed in this document, for more current and detailed information.

2 APPLICATION CONTAINER AND MICROSERVICES CHALLENGES

Application containers and microservices have unique characteristics with distinct security ramifications. Challenges have been identified around application container and microservices features through the lens of the Developer, Operator, and Architect (for microservices). To prioritize the challenges, the contributors developed a survey to score the challenges based on the methodology described below.

2.1 Survey to Score Application Container and Microservices Challenges

A survey to review each identified challenge across a defined set of weighted use case and feature questions was developed to prioritize said challenges. Contributors to this document applied the survey to each challenge, which resulted in the scored list of challenges in section 2.3 of this document.

2.2 Methodology

Public and private sector contributors collaboratively developed a list of questions to ask against each challenge. These questions were separated into system-specific use case and feature categories, then weighed by each member of the team. The following tables list the questions and their corresponding weight.

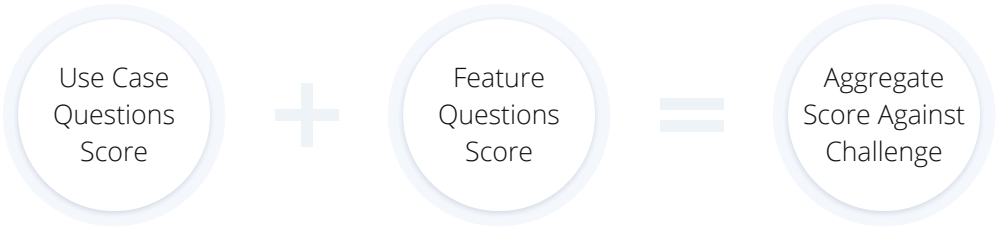
Table 1 - Use Case Questions

Is the container public facing?	50
If a container is compromised, will it result in lateral movement?	50

Table 2 - Feature Questions

Could the absence result in container image integrity compromise?	17
Could the absence result in critical data access?	17
Could this component's/function's/policy's weakness be used in privilege escalation?	16
Could the absence result in run-time compromise?	13
Could the absence result in container communication compromise?	12
Could this component's/function's/policy's weakness be used to override containers' configuration (capabilities/namespaces/cgroups)?	12
Could this component's/function's/policy's weakness result in the creation of a covert channel?	7
Could the absence result in non-compliance to any major security standards?	7

The use case questions and the weighted feature questions in the above tables were asked against each challenge, resulting in the prioritized list of scored challenges in section 2.3 of this document.



2.3 Challenges

The following tables present a list of challenges for application containers and microservices, with the challenges that scored highest listed first. Note that this is not a comprehensive list of challenges, but rather those that are the most prevalent at the time of writing.

The viewpoints reflected in these challenges are through the lens of the Developer, Operator, and Architect (for microservices) and are reflective of a vendor-agnostic approach to application containers and microservices.

Table 3 - Application Container Challenges

Application Container Challenges	Use Case / Feature	Viewpoint	Score
In multi-tenant hosting environments, a challenge exists to use monitoring and security solutions that do not require root or "privileged" access to the container host, as these solutions are specifically designed to expose information from more than one tenant. Compromise of monitoring tools could result in information spillage or system compromise.	3.1.3	Operator	10
Ensuring trust in host systems that store images and launch containers is a challenge. The absence of host-hardening processes can result in the compromise of images, runtime environments, and stored data.	3.1.8	Operator	10
Host hardening is an important requirement for secure container hosting, and a challenge exists to ensure that hardening does not interfere with authorized capabilities (e.g., network, storage) of the containers themselves. The absence can result in service availability issues.	3.1.2	Operator	9

Code and image provenance is a Developer and application owner concern; verification requires certification and key management across the code promotion path. This can present a challenge, as multiple parties and certificate owners may be involved. The absence can result in application version mismatches or running tampered images.	3.1.1	Developer	8
Containers on the same host must be firewalled from each other as well as connected networks. There is a challenge to ensure host-based firewalling provides network isolation for containers as well as protecting services on the host. The absence can result in information leakage or system compromise.	3.1.2	Operator	8
In some environments, there is a requirement for a container to only run on hosts which have attested security posture. A challenge exists to ensure applications and microservices are configured to run on approved computing infrastructure. The absence can result in failure to adhere to compliance regulations.	3.1.2	Developer	8
Ensuring the security of the container runtime—how and with whom it communicates—is critical and can be challenging to ensure in an environment where container hosts are dynamically created and managed on demand. The absence can result in information spillage and system compromise.	3.1.2	Operator	8
Monitoring network activity and providing the infrastructure to do so can present a challenge because of the lack of network tooling and data storage. The absence can result in diminished incident response and detection capabilities.	3.1.5	Operator	8
Ensuring that images are signed and validated as part of the build process can be a challenge due to differences in container management platform (CMP) features (see 3.1.6). The absence of this can result in unsigned or invalidated images.	3.1.8	Developer	8
Ensuring third-party components are free of vulnerabilities and updated as needed can be a challenge due to differing update frequencies by third-party vendors (see 3.1.6). The absence of this can result in the compromise of third-party components.	3.1.8	Developer	8

Ensuring only required components are packaged inside the image can be a challenge due to the complexity of applications (see 3.1.6). The absence of this can result in the compromise of an unnecessary component.	3.1.8	Developer	8
Building an image that minimizes using shared container volumes is a challenge due to the risk of exposing sensitive data to the host or other containers and current reliance to share information with containers. The absence can result in exposure of sensitive data.	3.1.9	Developer	8
Ensuring the ability to automate the configuration of the secret management tool and dynamically generate secrets required by the application in a Continuous Integration/Continuous Deployment (CI/CD) environment is a challenge because of the lack of best practices, tooling, and automation. The absence can result in the compromise of the application and/or data.	3.1.10	Operator	8
Ensuring the container microservice and runtime environment can securely consume and/or pass needed shared secrets is a challenge because of the lack of secure methods in deploying identifying keys for containers to retrieve application specific secrets. The absence of these secure methods could result in resource compromise, possibly due to secrets being saved in images themselves.	3.1.10	Operator	8
Importing and exporting container images and metadata without exposing sensitive data can be a challenge because of differences in CMPs, as well as ensuring that data from neighboring containers is not accidently handled. The absence can result in exposure of sensitive data.	3.1.16	Operator	8
Replicating security and container runtime configurations across different cloud service providers (CSPs) can be a challenge because of differences in CMPs and methods of security and authorization configuration. The absence can result in different configurations across CSPs, resulting in container compromise.	3.1.16	Developer	8
Activating a rehosted container (target) and removing all security configurations from the source container in a single atomic operation is a challenge because of differences in CMPs. The absence can result in exposure of sensitive security configurations.	3.1.16	Operator	8

Rehosting container groups while keeping security configurations and inter-container authorizations in place can be a challenge due to differences in CSP/CMP feature sets and implementations. The absence can result in differing security configurations resulting in container compromise.	3.1.16	Operator	8
Ensuring identity and access management (IAM) services are in place with authentication and authorization mechanisms is a challenge due to the variety of IAM services and their compatibility with containers. The absence can result in unauthenticated/unauthorized access to containers.	3.1.17	Developer	8
Ensuring credentials are authenticated can be a challenge due to differences in IAM services. The absence can result in unauthenticated/unauthorized access to containers.	3.1.17	Operator	8
Ensuring privileged user access and activities are logged and audited can be a challenge due the feature differences in CMPs. The absence can result in exposure of sensitive data.	3.1.17	Operator	8
Ensuring separation of duties can be a challenge due to feature differences in CMPs. The absence can result in exposure of sensitive data.	3.1.17	Operator	8
Providing a mechanism to protect against a CSP's malicious internal admin threats can be a challenge, as CSPs may not provide granular access to their logs. The absence can result in exposure of sensitive data.	3.1.17	Developer	8
Developers require easy and safe access of cryptographically verifiable container images. This creates a challenge for Operators to manage one or more trusted image repositories, depending on the number of environments. The absence can result in information spillage or system compromise.	3.1.1	Developer	7
Ensuring the same versions of an application running across multiple environments is a challenge because multiple parties, sites, and deployment paths may be involved. The absence can result in application version mismatches or issues with service availability.	3.1.1	Operator	7

In blue-green or canary-based deployment patterns, it can be challenging for the Developer to understand which container versions are running and where. This information needs to be easily and quickly processed by the appropriate parties. The absence will result in application version mismatches and system availability issues.	3.1.3	Developer	7
As the container lifecycle and local storage can both be short in duration, a challenge exists to ensure that audit logs from containers are gathered and stored securely for forensic use. The absence can result in diminished incident response and detection capabilities.	3.1.3	Operator	7
Since an application may be made up of a dynamic collection of containers, coalescing logs from these containers is a challenge as it requires more effort to reassemble and understand events. The absence can result in diminished incident response and detection capabilities.	3.1.3	Operator	7
Ensuring the isolation of the host interface from the container interface is a challenge due to a lack of existing container vendor support and an incompatible infrastructure. The absence can result in information spillage and system compromise.	3.1.4	Operator	7
Monitoring network activity and providing the infrastructure to do so is challenging due to a lack of existing container vendor support and an incompatible infrastructure, lack of access to the network. The absence can result in diminished incident response and detection capabilities.	3.1.4	Operator	7
Ensuring image signing and validation mechanisms are present is a challenge due to a lack of container vendor support. The absence of signing and validation can result in tampering and compromise.	3.1.6	Operator	7
Ensuring the automation and transparency of signing and validation needs can be a challenge due to lack of container vendor support and tooling. The absence of automation and transparency can lead to manual processes that are open to faults and tampering.	3.1.6	Operator	7

Ensuring the capability to investigate host, container, and other infrastructure logs for anomalies and possible signs of compromise (log forensics) is a challenge due to the lack of analysis tools and capabilities. The absence can result in diminishing incident response and detection capabilities.	3.1.7	Operator	7
Ensuring container runtime integrity is monitored and maintained via binary introspection, behavioral monitoring (e.g., system call, network, etc.) is a challenge due to the lack of monitoring and analysis tools, container vendor support, and technical skills. The absence can result in diminishing incident response and detection capabilities.	3.1.8	Operator	7
Implementing image repositories that support access control, image signing, and logging is a challenge due to the lack of infrastructure and resources. The absence of access control and image signing can result in the compromise of images and the runtime environment. The absence of logging related activities can result in diminishing incident response and detection capabilities.	3.1.8	Operator	7
Building an image that does not share the host filesystem is a challenge because exposing the host filesystem can result in compromising the underlying infrastructure and other containers that share the same environment. The ease of using this configuration is the actual challenge for the Developer.	3.1.9	Developer	7
Providing infrastructure that supports access control and monitoring of container volumes is a challenge because of the lack of existing container vendor support, incompatible infrastructure, and the requirement for privileged access by security solutions. The absence can result in information spillage and system compromise.	3.1.9	Operator	7
The dynamic deployment of certificates and key pairs to ensure Secure Sockets Layer (SSL) communication between containers or container to host is a challenge because of the lack of secure methods in enrolling certificates with the certificate authority.	3.1.10	Operator	7

Developing a containerized application that can be scaled up or down without notice is a challenge because of the extra effort required to design and implement the application in a manner where it can gracefully handle such scaling events while maintaining a secure state. The absence can result in a container terminating in an insecure state.	3.1.11	Developer	7
In addition to authorized resource usage, the Operator faces a challenge to ensure resources are not overused or abused, because this will affect usage of others in a shared environment. The absence may result in a DOS-style scenario.	3.1.12	Operator	7
When containers are imported into a trusted registry, a mechanism should ensure their configuration compliance against known baselines. This can be a challenge due to differences in CMP features. The absence of this capability can result in misconfiguration and container compromise.	3.1.16	Developer	7
Knowing the encryption algorithm used to encrypt the container can be a challenge due to feature differences in containers and CMPs. The absence can result in potentially unencrypted containers.	3.1.17	Developer	7
Ensuring the Key Management System (KMS) and its associated Application Program Interface (API) integrate with the key management services employed with containers can be a challenge due to feature differences in containers and CMPs. The absence can result in potentially unencrypted containers.	3.1.17	Developer	7
Ensuring user level entitlements to access keys from the KMS can be a challenge due to differences in CMP features. The absence can result in the inability for users to access encrypted containers.	3.1.17	Operator	7
Ensuring a secure communications transport exists between containers to ensure the confidentiality and integrity of data can be a challenge because of the lack of container vendor and OS support, along with network infrastructure limitations. The absence can result in information spillage and system compromise.	3.1.6	Operator	6

Ensuring coding policies include logging capabilities at the planning and design stages is a challenge due to the lack of presence of security concerns in the design phase and tooling. Usually, application logging is added after the fact as a response to an incident or audit. The absence can result in diminishing incident response and detection capabilities.	3.1.7	Developer	6
Ensuring application logs, starting with authentication logs, are captured for existing applications is a challenge due to lack of resources to modify the application and necessary guidance to do so. The absence can result in diminishing incident response and detection capabilities.	3.1.7	Developer	6
Providing infrastructure that supports encryption of data at rest is a challenge due to the lack of secret sharing standards and the exposure of data to third parties in multitenant environments. The absence can result in exposure of sensitive data.	3.1.9	Operator	6
Ensuring that there is no sensitive data (such as credentials) in the application code or in configuration files is a challenge because container runtime environments have varying security controls and may provide access to unintended parties. Developers need to consider new threats brought about by multi-tenancy and shared platforms. The absence can result in exposure of sensitive data.	3.1.10	Developer	6
Ensuring resource scaling requests take priority over other containers may provide a challenge to the Operator. This is due to complexity and implementation of the CMP to allow such prioritization. The absence can result in priority applications not being able to scale for user workloads.	3.1.14	Operator	6
Ensuring that the Operator has granular control over resource management can provide a challenge in a container environment. This is because such control requires a detailed understanding of each containerized application, as compared to managing the resources of virtual machines. The absence can result in resources incorrectly scaled for a container's needs.	3.1.14	Operator	6

Investigating network traffic related to containers by collecting and analyzing netflows and full packet capture is a challenge due to lack of network infrastructure and tooling (e.g., to trace a flow within a container to network related flows). The absence can result in diminishing incident response and detection capabilities.	3.1.7	Operator	5
Investigating binaries and related activities in runtime for possible compromise is a challenge due to the lack of monitoring tools and capabilities. The absence can result in an attacker compromising the runtime and tampering with the binaries to capture sensitive information such as credentials.	3.1.7	Operator	5
In some organizations, there may be a need to prioritize resource availability between applications. In these situations, the prioritization capabilities (or lack thereof) can provide a challenge to implementation of such functionality. The absence can result in the organization having to develop more complex infrastructure or experiencing system availability issues.	3.1.12	Developer	5
Ensuring that all resource requests can be met in a multi-tenant environment is a challenge to the Operator, who may not have the data to apply quality of service tiers to the application. The absence may result in a lack of available resources for one or more applications or tenants.	3.1.12	Operator	5
The Operator faces a challenge to ensure that the CMP balances resource consumption across users and applications as defined by quotas. This can be a challenge, as the scales used for resource definition in application containers are more fine-grained than those used for bare or virtual machines and can require more effort to reach a balance where resources are efficiently used. The absence can result in resources being misallocated within the system.	3.1.13	Operator	5
Due to the transient nature of containers, the Operator faces a challenge if attempting to back up data contained within the container. This is because the container can exit (and be destroyed) before backup can be performed. The absence can result in critical data not being backed up.	3.1.15	Developer	4

Ensuring resource availability as needed to support workloads is a challenge to the Developer. This is because of the dynamic use and availability of resources in a potentially shared environment, combined with the possibly unpredictable use of an application. The absence can result in either the CMP being unable to allocate appropriate resources for an application container, or a resource shortage within the CMP.	3.1.12	Developer	4

Table 4 - Microservices Challenges

Microservices Challenges	Viewpoint	Score
Architects will be challenged to find a balance between the costs and benefits of rebuilding to a microservice architecture and then orchestrating those microservices. The absence will result in either cost overruns or an application that does not fully benefit from a microservice architecture.	Architect	10
The container lifespan provides a challenge when architecting stateful microservices. Care must be taken to design a microservice that does not lose state when a container is no longer running. The absence can result in data loss.	Architect	10
A Developer writing or maintaining a microservice that interfaces with several other microservices may be challenged to ensure interoperability. Reliable test harnesses are required. The absence can result in increased availability and performance issues as the application is decomposed into microservices.	Developer	10
Ensuring that a microservice architecture provides a means for authentication and authorization of access to a microservice can be a challenge due to the variety of IAM solutions and their compatibility with the hosting architecture of the microservice. The absence of this can result in a DOS scenario resulting from lack of authentication methods.	Architect	10
The container lifespan provides a challenge when architecting stateful microservices. Care must be taken to design a microservice that does not lose state when a container is no longer running.	Architect	10

As an application is broken down into various components, there is a challenge to provide a proven code promotion system for use in the development process and through staging and production. The absence of this can result in increased delays in code promotion.	Operator	9
Microservices have a higher requirement for secure communication between components than that found in legacy applications. Operators can be challenged to provide a flexible secure network transport between selected microservices. The absence can result in information spillage and system compromise.	Operator	9
In a microservices architecture, each microservice needs to be autonomous. This can be a challenge for a Developer used to tightly-bound monolithic applications. The absence can result in performance issues or lack of availability.	Developer	9
To support scaling and availability requirements, microservices are frequently run in multiples across several systems. It is a challenge for the Developer to write such services in a manner so that multiple copies can run at the same time and co-exist. The absence of this can result in data destruction or availability issues for the microservice.	Developer	9
As an application gets broken down into its components, there is a challenge to provide a proven code promotion system, for use in the development process and through staging and production. The absence of this can result in increased delays in code promotion.	Operator	9
Microservices have a higher requirement for secure communication between components than that found in legacy applications. Operators can be challenged to provide a flexible secure network transport between selected microservices. The absence can result in information spillage and system compromise.	Operator	9
As an application becomes more fragmented, the Developer has less visibility into microservices. This creates a challenge to ensure each component works with the others. The absence can result in data spillage, data destruction, and availability issues.	Developer	8

When consuming a microservice, either a synchronous or an asynchronous protocol may be used, depending on several factors of the microservice. For Developers, this can provide a challenge to keep track of which services are consumed in which manner. The absence can result in delays in the development lifecycle and data destruction.	Developer	8
The focus of a microservice on a single deliverable will challenge Developers to write highly cohesive and secure objects. The absence can result in information spillage, system compromise, or availability issues.	Developer	8
As applications become more fragmented, Developers will be challenged to abstract their code to higher levels, resulting in the use of common libraries. The absence can result in inconsistency of use cases, destruction of data, and availability issues.	Developer	8
Microservices have a requirement for strong monitoring. Operators can be challenged to provide services that scale with the microservice environment. The absence can result in availability issues.	Operator	8
Architects will be challenged to balance the cohesion needs of the microservice with the goal of reusable code. The absence can result in delayed development lifecycles and code promotion, along with reliability and consistency issues in production.	Architect	8
Architects will be challenged to include the needs of signing authorities and certificates in their planning. The absence can result in information spillage, system compromise, or system availability issues.	Architect	8
Trusted signing authorities need to be established and communicated as microservices are adopted. The absence can result in inconsistency of use cases, destruction of data, and availability issues.	Operator	7
A Developer writing or maintaining a microservice will be challenged with the need to ensure the integrity of the microservice. The absence can result in data loss, information spillage, or system compromise.	Developer	7

3. APPLICATION CONTAINER AND MICROSERVICES: USE CASES AND FEATURES

The following table consists of use cases and features for application containers and microservices:

Table 5 - ACM / Microservices Use Cases and Features

Title	Use Case / Feature	Description
Code Promotion Across Environments (Development, Quality Assurance, Test, Production)	3.1.1	Ensure Developers and Operators are equally engaged in the process of code promotion across application container environments.
Securing the Host	3.1.2	Ensure container host is hardened and has minimal exposure to other containers yet allows authorized guest containers to operate.
Container Continuous Monitoring from the Platform/Host	3.1.3	Ensure the container runtime actions and events are monitored and logged.
Container Networking - Communications between Host and Container	3.1.4	Ensure communications between host and containers are secured, network traffic is monitored, and access to the network traffic, resources, and configurations is controlled.
Container Networking - Communications between Containers	3.1.5	Ensure communications between containers are secured, network traffic is monitored, and access to the network traffic, resources, and configurations is controlled.
Validate Integrity & Security Quality of the Image	3.1.6	Ensure images have no known vulnerabilities or malicious code. Ensure images have not been tampered with as they move from development to production.
Container Forensics	3.1.7	Ensure the container infrastructure supports digital forensics capabilities that can be integrated into an incident response plan.

Trust Chain Through Containers	3.1.8	Ensure images and containers have not been tampered with at rest, in transport, and through runtime as they move from development to production.
Container Volume Management	3.1.9	Ensure the container has need-only access to data volumes and uses dedicated storage resources for runtime operations.
Container Secret Management	3.1.10	Ensure sensitive data such as API keys, passphrases, or database credentials are not present in container images
Platform Management - Notification of Lifecycle Events	3.1.11	Ensure stateful containerized applications are aware of container scaling events to ensure transitions between secure states.
Platform Management - Resource Request	3.1.12	Ensure resource availability as needed for dynamic (possibly multi-tenant) workloads. If possible, provide support for request prioritization.
Platform Management - Container Resource Management	3.1.13	Ensure an understanding of the resource requirements of the containerized application; work with Operators to appropriately define and use quotas.
Container Management - Scaling Container Resources	3.1.14	Ensure ability to scale application containers up and down; provide granular control over resource management.
Container Management - Data Backups and Replication	3.1.15	Ensure ability to back up data in potentially transitory application containers before their destruction.
Container Management - Container Rehosting between CMPs	3.1.16	When containerized applications are rehosted, ensure security configurations are migrated appropriately.
Container Encryption	3.1.17	Ensure the ability to encrypt configuration, data, and (some) applications within the container image.

Application Decomposition into Microservices	———	Refactor existing applications into component microservices. Microservices should be built around capabilities and should have the ability to operate independently.
New Application Development using a Microservices Architecture	———	Create a new application using a microservices architecture. Employment of a microservices architecture allows for continuous integration and continuous deployment, key elements of the DevOps cycle.
Microservice Integrity Validation	———	Ensure internal integrity and integrity validation across microservices. Microservices that fail integrity validation will not be able to operate effectively or may introduce an attack vector that can result in application/data compromise.

3.1 Description of Use Cases and Features

The following sections further describe the aforementioned use cases and features of application containers and microservices.

3.1.1 Code Promotion Across Environments (Development, Quality Assurance, Test, Production)

Regarding code promotion across environments, Developers and Operators need to be equally engaged in the process of code promotion across application container environments. Today, Developers can push code across application container environments without the need for Operator interaction.

Encountered or perceived challenges to code promotion across environments between Developer and Operator are as follows:

Developer's viewpoint:

1. Code and image provenance is a Developer and application owner concern; verification requires certification and key management across the code promotion path. This can be a challenge, as multiple parties and certificate owners may be involved. The absence can result in application version mismatches or running tampered images.

Operator's viewpoint:

1. Developers require easy and safe access of cryptographically verifiable container images. This creates a challenge for Operators to manage one or more trusted image repositories depending on the number of environments. The absence can result in information spillage or system compromise.
2. Ensuring the same versions of an application running across multiple environments is a challenge because multiple parties, sites, and deployment paths may be involved. The absence will result in application version mismatches or issues with service availability.

3.1.2 Securing the Host

When securing the host, one must ensure the application container host is hardened and has minimal exposure to containers, yet also allows authorized guest containers to operate. Securing the host should be considered a standard organizational process [NIST SP 800-123], possibly with some modifications for the container workload.

Encountered or perceived challenges to securing the host include the following viewpoints:

Developer's viewpoint:

1. No Developer viewpoints exist for this challenge.

Operator's viewpoint:

1. Ensuring equivalent security functions for containers across a cluster of hosts can present a challenge because container orchestration systems need to seamlessly automate security functions and their configuration. The absence can result in information spillage, availability issues, or system compromise.
2. In some environments, there is a requirement for a container to only run on hosts that can attest to their security posture. A challenge exists to automate and provide this attestation information to the container in a timely fashion. The absence can result in failure to attest to compliance regulations.
3. Host hardening is an important requirement for secure container hosting, but a challenge exists to ensure that hardening does not interfere with authorized capabilities (e.g., network or storage) of the containers themselves. The absence can result in service availability issues.
4. Containers on the same host must be firewalled from each other as well as from connected networks. There is a challenge to ensure that host-based firewalling provides network isolation for containers and that it protects services on the host. The absence can result in information leakage or system compromise.
5. Ensuring the security of the container runtime—how and with whom it communicates—is critical and can be challenging to ensure in an environment where container hosts are dynamically created and managed on demand. The absence can result in information spillage and system compromise.

3.1.3 Container Continuous Monitoring from the Platform/Host

The ephemeral nature of application containers increases the necessity of reliable centralized monitoring of platform, host, and containers. In this instance, one must ensure the container runtime actions and events are monitored and logged.

Encountered or perceived challenges for this instance include the following viewpoints:

Developer's viewpoint:

1. In blue-green or canary-based deployment patterns, there may be a challenge for the Developer to understand which container versions are running and where. This information needs to be easily and quickly digestible by appropriate parties. The absence can result in application version mismatches and system availability issues.

Operator's Viewpoint:

1. As local storage and the container lifecycle may be short in duration, a challenge exists to ensure that logs from containers are gathered and stored securely for forensic use. The absence can result in diminished incident response and detection capabilities.
2. As an application may be made up of a dynamic collection of containers, coalescing logs from these containers is a challenge requiring more effort to reassemble and understand events. The absence can result in diminished incident response and detection capabilities.
3. In multi-tenant hosting environments, there exists a challenge to use monitoring and security solutions that do not require root or "privileged" access to the container host, as these solutions generally would expose information from more than one customer. The absence may result in information spillage or system compromise.

3.1.4 Container Networking - Communications Between Host and Container

In container networking, it is important to ensure communications between host and containers are secured, network traffic is monitored, and access to the network traffic, resources and configurations is controlled.

Encountered or perceived challenges to communications between host and container include the following viewpoints:

Developer's viewpoint:

1. Host-to-container networking will be transparent for Developers.

Operator's viewpoint:

1. Ensuring the ability of the container and CMP to provide multi-tenancy, separation of resources, and to mitigate container escape attack is a challenge because of the lack of existing container vendor support and an incompatible infrastructure. The absence can result in information spillage and system compromise.

2. Ensuring the isolation of the host interface from the container interface is a challenge because of the lack of existing container vendor support and an incompatible infrastructure. The absence can result in information spillage and system compromise.
3. Monitoring network activity and providing the infrastructure to do so is challenging because of the lack of existing container vendor support, an incompatible infrastructure, and lack of access to the network. The absence can result in diminished incident response and detection capabilities.

3.1.5 Container Networking - Communications Between Containers

In container networking, communications between containers must be secured, network traffic is monitored, and access to the network traffic, resources, and configurations is controlled. In terms of application, communications between containers affect access control—limiting access to network traffic, resources, and configurations (leveraging network namespace). Other outcomes include isolating the container interfaces, monitoring network activity, and securing communications between containers via SSL/TLS or VPN.

Encountered or perceived challenges to communications between containers include the following viewpoints:

Developer's viewpoint:

1. Certain network configurations that may enhance security may create challenges with simultaneously allowing all expected communications between containers.

Operator's viewpoint:

1. Limiting access to container network traffic, resources, and configurations can be a challenge because lack of operating system (OS) controls and container vendor support. Misconfiguration of access can result in information spillage and system compromise.
2. The isolation of the host interface from the container interface can be a challenge because of the lack of OS and container networking tools. The absence can result in information spillage and system compromise.
3. Monitoring network activity and providing the infrastructure to do so can be a challenge because of the lack of network tooling and data storage. The absence can result in diminished incident response and detection capabilities.

3.1.6 Validate Integrity and Security Quality of the Image

In this instance, images must have no known vulnerabilities or malicious code. One must ensure images have not been tampered with as they move from development to production. Applications of this feature include scanning images for vulnerabilities and malicious code and using image signing and validation mechanisms.

Encountered or perceived challenges to validating the integrity and security quality of the image include the following viewpoints:

Developer's viewpoint:

1. Signing and validation of images can be a challenge due to the lack of container vendor support and lack of tooling. The absence of signing and validation can result in image tampering and compromise.
2. Using and maintaining up-to-date libraries can be a challenge due to the variety of third-party tools and their corresponding update frequency. The absence can increase the attack surface and risk of compromise.

Operator's viewpoint:

1. Ensuring image signing and validation mechanisms are present is a challenge due to the lack of container vendor support. The absence of signing and validation can result in tampering and compromise.
2. Ensuring the image does not contain vulnerabilities and malicious code can be a challenge due to the lack of detection tools and supporting infrastructure. The absence of ensuring this can result in system compromise.
3. Ensuring the automation and transparency of signing and validation needs can be a challenge due to lack of container vendor support and tooling. The absence of automation and transparency can lead to manual processes that are open to faults and tampering.
4. Ensuring that a secure communications transport exists between containers to ensure the confidentiality and integrity of data can be a challenge because of the lack of container vendor and OS support as well as network infrastructure limitations. The absence can result in information spillage and system compromise.

3.1.7 Container Forensics

This feature ensures the container infrastructure supports digital forensics capabilities that can be integrated into an incident response plan. Applications include image forensics (i.e., investigating the contents of an image), network forensics (i.e., investigating network traffic related to containers by collecting and analyzing netflows and full packet capture), memory forensics (i.e., investigating binaries and related activities in runtime for possible compromise), and log forensics (i.e., investigating host, container, and other infrastructure logs for anomalies and possible signs of compromise). It also ensures the infrastructure collects and provides access to the necessary data sources.

Encountered or perceived challenges to container forensics include the following viewpoints:

Developer's viewpoint:

1. Ensuring coding policies include logging capabilities at the planning and design stages is a challenge due to the lack of presence of security concerns in the design phase and tooling. Usually application logging is added after the fact as a response to an incident or audit. The absence will result in diminishing incident response and detection capabilities.
2. Ensuring application logs, starting with authentication logs, are captured for existing applications is a challenge due to lack of resources to modify the application and necessary guidance to do so. The absence will result in diminishing incident response and detection capabilities.

Operator's viewpoint:

1. Ensuring the potential to investigate the contents of an image (static analysis) is a challenge due to undocumented image formats, a lack of tooling, and infrastructure. The absence of this capability can result in diminished incident response and detection.
2. Investigating network traffic related to containers by collecting and analyzing netflows as well as full packet capture is a challenge due to lack of network infrastructure and tooling (e.g., to trace a flow within a container to network related flows). The absence will result in diminishing incident response and detection capabilities.
3. Investigating binaries and related activities in runtime for possible compromise is a challenge due to the lack of monitoring tools and capabilities. The absence can result in an attacker compromising the runtime and tampering with the binaries to capture sensitive information such as credentials.
4. Ensuring the capability to investigate host, container, and other infrastructure logs for anomalies and possible signs of compromise (log forensics) is a challenge due to the lack of analysis tools and capabilities. The absence will result in diminishing incident response and detection capabilities.

3.1.8 Trust Chain Through Containers

In trust chains through containers, images and containers have not been tampered with at rest, in transport, and in runtime as they move from development to production. In an application setting, one must ensure host systems that store images and launch containers are trusted, that the container engine is configured in a secured manner, that access control mechanisms are in place, image integrity and security quality is maintained (see 3.1.6), all communications are encrypted and mutually authenticated, and container runtime integrity is monitored and maintained.

Encountered or perceived challenges to the trust chain through containers include the following viewpoints:

Developer's viewpoint:

1. Ensuring that images are signed and validated as part of the build process can be a challenge due to differences in CMP features (see 3.1.6). The absence of this can result in unsigned or unvalidated images.
2. Ensuring third-party components are free of vulnerabilities and updated as needed can be a challenge due to differing update frequencies by third-party vendors (see 3.1.6). The absence of this can result in the compromise of third-party components.
3. Ensuring only required components are packaged inside the image can be a challenge due to the complexity of applications (see 3.1.6). The absence of this can result in the compromise of an unnecessary component.

Operator's viewpoint:

1. Ensuring container runtime integrity is monitored and maintained via binary introspection, behavioral monitoring (system call, network, etc.) is a challenge due to the lack of monitoring and analysis tools, container vendor support, and technical skills. The absence can result in diminishing incident response and detection capabilities.
2. Ensuring trust in host systems that store images and launch containers is a challenge due to the lack of policies, infrastructure, and other resources. The absence of host hardening processes can result in the compromise of the images, runtime environments and stored data.
3. Ensuring the secureness of image transactions with repositories that have access control mechanisms is a challenge due to the lack of role definitions, environment separation, and policies. The absence can result in information spillage.
4. Implementing image repositories that support access control, image signing, and logging is a challenge due to the lack of infrastructure and resources. The absence of access control and image signing can result in the compromise of the images and the runtime environment. The absence of logging of related activities will result in diminishing incident response and detection capabilities.
5. Implementing image validation mechanisms to ensure only authorized personnel and services can contribute and use images is a challenge due to the lack of supporting infrastructure, role definitions, and policies. The image validation can result in the compromise of the images and the runtime environment.

3.1.9 Container Volume Management

In container volume management, one must ensure the container has need-only access to data volumes and uses dedicated storage resources for runtime operations. In the application, one must ensure access control exists for file system resources and raw storage devices, that data is encrypted at rest in the data volumes, and one must avoid directly mounting a host directory.

Encountered or perceived challenges to container volume management include the following viewpoints:

Developer's viewpoint:

1. Building an image that minimizes using shared container volumes is a challenge due to the risk of exposing sensitive data to the host or other containers and current reliance to share information with containers. The absence can result in exposure of sensitive data.
2. Building an image that does not share the host filesystem is a challenge because exposing the host file system can result in the compromise of the underlying infrastructure and other containers sharing the same environment. The ease of using this configuration is the actual challenge for the Developer.

Operator's viewpoint:

1. Providing infrastructure that supports access control and monitoring of container volumes is a challenge because of the lack of existing container vendor support, incompatible infrastructure,

and the requirement for privileged access by security solutions. The absence can result in information spillage and system compromise.
2. Providing infrastructure that supports encryption of data at rest is a challenge due to the lack of secret sharing standards and the exposure of data to third parties in multi-tenant environments. The absence can result in exposure of sensitive data.

3.1.10 Container Secret Management

Container secret management ensures sensitive data such as API keys, passphrases, database credentials, and related components are not present in container images. In an application setting, sensitive data should be generated at runtime, transferred to the container in a secure manner, and stored encrypted.

Encountered or perceived challenges to container secret management include the following viewpoints:

Developer's viewpoint:

1. Ensuring that there is no sensitive data (such as credentials) in the application code or in configuration files is a challenge because the environments that containers run in have varying security controls and may provide access to unintended parties. Developers need to consider the new threats brought about by multi-tenancy and shared platforms. The absence can result in exposure of sensitive data.

Operator's viewpoint:

1. Ensuring the ability to automate the configuration of the secret management tool and dynamically generate secrets required by the application in a Continuous Integration/ Continuous Deployment (CI/CD) environment is a challenge because of the lack of best practices, tooling and automation for "key zero" (the first key in the key management system that is usually handled manually). The absence can result in the compromise of the application and/or data.
2. Ensuring that the container microservice and runtime environment has the ability to securely consume and or pass needed shared secrets is a challenge due to the lack of secure methods in deploying identifying keys for containers to retrieve application-specific secrets. The absence of these secure methods can result in resource compromise due to secrets being saved in images themselves.
3. The dynamic deployment of certificates and key pairs to ensure SSL communication between containers or container to host is a challenge because of the lack of secure methods in enrolling certificates with the certificate authority.

3.1.11 Platform Management - Notification of Lifecycle Events

In an ideal world, application containers are stateless, or the applications have been developed with clustering technologies to attach and release neighbors as they start and are terminated. Some use cases, though, will have containers that have not been developed in such a manner or contain legacy applications that have migrated to containers. Such applications running inside a container may not

be aware of pending lifecycle events issued by a CMP such as container start/stop/scale. The containerized application should have an opportunity to be informed of the pending container's lifecycle events to manage its own lifecycle accordingly. This is essential to ensure the application's secure startup and shutdown.

In an application scenario, notification of lifecycle events releases platform resources for terminated containers and maintains confidentiality of container data post-destruction.

Encountered or perceived challenges to the notification of "container lifecycle events" include the following viewpoints:

Developer's viewpoint:

1. Developing a containerized application that may be scaled up or down without notice is a challenge because of the extra effort required to design and implement the application in such a way that it can gracefully handle scaling events while maintaining a secure state. The absence can result in a container terminating in an insecure state.
2. Containerizing legacy applications might not render a desired effect and can result in a container terminating in an insecure state.

Operator's viewpoint:

1. CMPs and application containers are intended to be dynamic and scalable. This is a challenge for the Operator, where some application containers running legacy applications may not be as dynamic as expected, resulting in application crashes or insecure states.

   The Operator's activities, as we know them today, cannot compensate for the application deficiencies.

3.1.12 Platform Management - Resource Request

In computing environments such as microservice deployments, resource allocation (i.e., CPU, memory, disk I/O, network, etc.) and contention issues can arise due to the elasticity of workload and dynamic resource requests.

Encountered or perceived challenges to resource requests within platform management include the following viewpoints:

Developer's viewpoint:

1. No Developer viewpoints exist for this specific challenge

Operator's viewpoint:

1. Ensuring predictable platform behavior under conditions of varying resource availability can be a challenge, an Operator needs to evaluate setting resource constraints to preserve environment stability when resources availability fluctuates.

3.1.13 Platform Management - Container Resource Management

Containers could consume a very large amount of resources if auto-scaling capabilities are not properly constrained. This could result in a denial of service to containers on the same platform. It can be challenging to accurately estimate resource requirements before provisioning of infrastructure.

Encountered or perceived challenges to container resource management within platform management include the following viewpoints:

Developer's viewpoint:

1. No Developer viewpoints exist for this specific challenge

Operator's viewpoint:

1. The Operator faces a challenge to ensure that the CMP optimally balances resource consumption across users and applications as defined by quotas. This can be a challenge as the scales used for resource definition in application containers are more fine-grained than those of bare or virtual machines and may require more effort to reach a balance where resources are efficiently used. The absence can result in resources being misallocated within the system.

3.1.14 Container Management - Scaling Container Resources

It is a challenge to determine requirements to increase or decrease container resources.

Encountered or perceived challenges to scaling container resources include the following viewpoints:

Developer's viewpoint:

1. It is a challenge for Developers to define application performance profiles. Predictive application workload modeling is a complex task often substituted by a "common practice" or, equally precise, "by analogy," largely due to lack of tooling in development environments. Both are likely to lead to inefficient resource utilization.

Operator's viewpoint:

1. The Operator's challenge is to ensure optimal application performance at any given time. Attempts to accommodate the Developer's annotated resource requests might adversely affect the current application mix. Ensuring that the Operator has granular control over resource management is a necessity. Such control requires detailed understanding of each containerized application, as compared to managing the resources of virtual machines. The absence can result in resources incorrectly scaled for a container's needs.

3.1.15 Container Management - Data Backups and Replication

The paradigm for data backup and restores have some unique challenges compared to traditional virtual machines or bare metal. Containers are intended to be ephemeral. Containers use data volumes, which are directories that can be mapped to a medium such as a traditional storage device. Using volumes, a new container can be started with the existing dataset when a container dies.

Encountered or perceived challenges to data backups and replication include the following viewpoints:

Developer's viewpoint:

1. Properly documenting the requirements for an application backup and restore of an application can be a challenge. This documentation must include the state for which an application can be backed up and restored. For some applications, a simple snapshot of the storage medium may suffice, while for other applications, application-specific tools will need to be run. If an application needs to be taken offline or drained of active users/session in any capacity, then it must be documented.
2. If an application needs to be taken offline for a backup and/or restore, documenting what services flow upstream downstream from that application is a challenge.
3. It is a challenge to build service dependency retries and idempotency into the application.

Operator's viewpoint:

1. The Operator will implement how an application will be backed up and restored on a system using the criteria set forth by the Developer and organizational wide policies. These policies may include requirements for the number or frequency of the backups.
2. For simple use cases in which the storage snapshots will be sufficient, the Operator should schedule backups in accordance with the application and organizational requirements.
3. Some environments may include the ability to create volumes and dynamically map them to storage devices; this is often called dynamic provisioning. If an Operator is using a snapshot policy, then making this work with dynamic provisioning may not be possible since many storage mediums will not allow to set snapshot schedules until after the volume is created. If this cannot be done properly, an Operator may need to disable dynamic provisioning.
4. For non-simple use cases, the Operator will be responsible for creating the safe-state for which the application can be backed up and for running the application-specific tools required to back up the application. This may include standing up temporary application instances to handle the workload for application dependencies when the application is taken offline. This may be non-trivial. As an example, an Operator may need to:
    a. Roll out a blue-green deployment to offload traffic while the application is taken offline.
    b. Create the safe-state for backup/restore as listed by the Developer and/or vendor of the application. This may include draining sessions or closing out all transactions. Implementing this in a container scenario could involve using a sidecar pattern (an agent running a co-located container next to a database) or using other cron tools.

c. Running the application-specific tool is required to back up the application. As an example, for web servers this may include replication all events to a secondary cluster. For databases this may include running specific tools. If the tool creates an archive of data, then the data needs to be backed up safely to a storage medium. This may require custom scripting to move the data archive to a safe location.

3.1.16 Container Management - Container Rehosting between CMPs

When migrating containers between CMPs, container security needs to be considered. Base security evaluations of the provider should be addressed internally. In addition, container platform Developers and administrators need to be aware of the differences between CMPs for configuring the container platform itself and container Developers and tenants should be isolated from the impact of container rehosting.

Encountered or perceived challenges to container rehosting between CMPs include the following viewpoints:

Developer's viewpoint:

1. No Developer viewpoints exist for this challenge.

Operator's viewpoint:

1. When containers are imported into a trusted registry, a mechanism should ensure their configuration compliance against known baselines. This can be a challenge due to differences in CMP features. The absence of this capability can result in misconfiguration and container compromise.
2. Importing and exporting container images and metadata without exposing sensitive data as well as ensuring that data from neighboring containers is not accidently handled can be a challenge because of differences in CMPs. The absence can result in exposure of sensitive data.
3. Replicating security and container runtime configurations across different CSPs can be a challenge because of differences in CMPs and methods of security and authorization configuration. The absence can result in different configurations across CSPs, resulting in container compromise.
4. Activating a rehosted container (target) and removing all security configurations from the source container in a single atomic operation is a challenge because of differences in CMPs. The absence can result in exposure of sensitive security configurations.
5. Rehosting container groups while keeping security configurations and inter-container authorizations in place can be a challenge due to differences in CSP/CMP feature sets and implementations. The absence can result in differing security configurations, which can lead to container compromise.

3.1.17 Container Encryption

Configuration of data at rest in container environments is susceptible to threats like those experienced in traditional data center environments, including threats from external attackers and malicious insiders within CSPs. Container images often have sensitive credentials injected at runtime, usually in the form of secrets, that will need to be protected. Additionally, there could be sensitive data produced by the

container applications that will need to be stored a volume, and that volume may need to be encrypted depending on the agency accreditation, organization policies, and regulatory compliance. Encryption needs to always be implemented for data in transit as well as for data at rest for secrets and other sensitive data. Additional requirements for data in motion, while rare, may be required for some environments with high security postures.

Encountered or perceived challenges to container encryption include the following viewpoints:

Developer's viewpoint:

1. Sensitive configurations or data must not be hard coded or built into the container image itself in any way. This will require best practices for image builds to be developed by the organization and implemented by the Developer.
2. Ensuring that sensitive data is not written to log files.
3. Ensuring that sensitive data is not printed out for debugging or other error reporting.
4. Ensuring that sensitive data is not printed to user facing pages, even while in a debug mode, since if the debug mode is accidentally left on it may comprise sensitive information.
5. Documenting any data that will need to be injected in a container at runtime. This will include denoting which data may be sensitive.
6. Implementing encryption of data in transit through the use of PKI or other encryption methods. The Operator may also implement this using tools such as IPSec.
7. Documenting any known dependencies on the operating system for which the container will be running. As an example, if an application attempts to use FIPS 140-2 validated modules from an operating system then it will need to be fully documented so an Operator can ensure that proper environment is provided.
8. Documenting and possibly implementing any additional requirements such as encryption of data in motion or the requirement for third-party tools such as a key management system.

Operator's viewpoint:

1. Ensuring that least privilege is followed by only allowing credentials sensitive data to be injected or mounted to containers that will require the data.
2. Ensuring that least privilege is followed by only allowing nodes that will run containers with sensitive data to have access those secrets or other sensitive data sets.
3. Ensuring the use of encryption of data in transit. This may include turning on IPSec for all container communications and/or ensuring that the Developer has followed best practices by encrypting all container network communications.
4. Ensuring the proper environment is provided to meet container application encryption requirements. An example may be providing a machine with particular validated FIPS 140-2 crypto modules.
5. Ensuring credentials are authenticated can be a challenge due to differences in IAM services. The absence can result in unauthenticated/unauthorized access to containers.
6. Ensuring privileged user access and activities are logged and audited can be a challenge due to the feature differences in CMPs. The absence can result in exposure of sensitive data.
7. Ensuring separation of duties can be a challenge due to feature differences in CMPs. The absence can result in exposure of sensitive data.

8. Providing a mechanism to protect against CSPs' malicious internal admin threats can be a challenge as CSPs may not provide granular access to their logs. The absence can result in exposure of sensitive data.
9. Ensuring user-level entitlements to access keys from the KMS can be a challenge due to differences in CMP features. The absence can result in the inability for users to access encrypted containers

4 MICROSERVICES

4.1 Microservices: Use Cases

The following descriptions outline the use cases within microservices.

4.1.1 Application Decomposition into Microservices

In application decomposition, Developers need to refactor existing applications into component microservices. In an application sense, restructuring applications into microservices will allow for higher velocity changes.

Encountered or perceived challenges to application decomposition into microservices include the following viewpoints:

Developer's viewpoint:

1. As an application becomes more fragmented, the Developer has less visibility into microservices. This creates a challenge to ensure each component works flawlessly with others. The absence can result in data spillage, data destruction, and availability issues.
2. A Developer writing or maintaining a microservice that interfaces with several other microservices may be challenged to ensure interoperability. Reliable test harnesses are required. The absence can result in increased availability and performance issues as the application is decomposed into microservices.

Operator's viewpoint:

1. As an application is broken into various components, there is a challenge to provide a proven code promotion system for use in the development process and through staging and production. The absence of this can result in increased delays in code promotion.
2. Microservices have a higher requirement for secure communication between components than those found in legacy applications. Operators can be challenged to provide a flexible, secure network transport between selected microservices. The absence will result in information spillage and system compromise.

Architect's viewpoint:

1. Architects will be challenged to find a balance between the costs and benefits of rebuilding to a microservice architecture and then orchestrating those microservices. The absence can result in either cost overruns or an application that does not fully benefit from a microservice architecture.
2. The container lifespan provides a challenge when architecting stateful microservices. Care must be taken to design a microservice that does not lose state when a container is no longer running. The absence will result in data loss.

4.1.2 New Application Development Using a Microservices Architecture

Developers need to create a new application using a microservices architecture. Microservices architecture allows multiple presentation components to integrate seamlessly with business logic and databases while allowing high-velocity changes in a continuous deployment environment.

The encountered or perceived challenges to new application development using a microservices architecture include the following viewpoints:

Developer's viewpoint:

1. In a microservices architecture, each microservice needs to be autonomous. This can be a challenge for a Developer used to tightly-bound monolithic applications. The absence can result in performance issues or lack of availability.
2. When consuming a microservice, either a synchronous or an asynchronous protocol may be used depending on several factors of the microservice. For Developers, this may provide a challenge to keep track of how certain services are consumed. The absence can result in delays in the development lifecycle and data destruction.
3. To support scaling and availability requirements, microservices are frequently run in multiples across several systems. Developers are challenged to write such services in a manner such that multiple copies can run at the same time and coexist. The absence of this can result in data destruction or availability issues for the microservice.

Operator's viewpoint:

1. As an application is broken down into its components, there is a challenge to provide a proven code promotion system for use in the development process and through staging and production. The absence of this can result in increased delays in code promotion.
2. Microservices have a higher requirement for secure communication between components than those found in legacy applications. Operators can be challenged to provide a flexible, secure network transport between selected microservices. The absence can result in information spillage and system compromise.

Architect's viewpoint:

1. The transient nature of containers provides a challenge when architecting stateful microservices. Care must be taken to design a microservice that does not lose state when a container is no longer running.
2. Ensuring a microservice architecture provides means for authentication and authorization of access to a microservice can be a challenge due to the variety of IAM solutions and their compatibility with the hosting architecture of the microservice. The absence of this can result in a DOS scenario resulting from lack of authentication methods.

   As microservices each provide smaller, usually quicker services that are accessed more often, they require authentication and authorization functionalities that also are efficient in nature. Otherwise, as the number of microservices scales, the amount of time spent authenticating and authorizing the caller will increase significantly.

4.2 Microservices: Features

The following descriptions outline features within microservices.

4.2.1 Microservice Integrity Validation

Microservices need to ensure integrity both internally and across the deployed microservices. In an application sense, authoritative sources for signing and certificate management are required to ensure a secure environment.

Encountered or perceived challenges to microservice integrity validation include the following viewpoints:

Developer's viewpoint:

1. A Developer writing or maintaining a microservice will be challenged with the need to ensure the integrity of the microservice. The absence can result in data loss, information spillage, or system compromise.
2. The focus of a microservice on a single deliverable will challenge Developers to write highly cohesive and secure objects. The absence can result in information spillage, system compromise, or availability issues.
3. As applications become more fragmented, the Developer will be challenged to abstract the code to higher levels, resulting in the use of common libraries. The absence can result in inconsistency of use cases, destruction of data, and availability issues.

Operator's viewpoint:

1. Trusted signing authorities need to be established and communicated as microservices are adopted. The absence can result in inconsistency of use cases, destruction of data, and availability issues.
2. Microservices have a requirement for strong monitoring. Operators can be challenged to provide services that scale with the microservice environment. The absence can result in availability issues.

Architect's viewpoint:

1. Architects will be challenged to balance the cohesion needs of the microservice with the goal of reusable code. The absence can result in delayed development lifecycles and code promotion, along with reliability and consistency issues in production.

   Architects will be challenged to include the needs of signing authorities and certificates in their planning. The absence will result in information spillage, system compromise, or system availability issues.

Appendix A–Acronyms

Selected acronyms and abbreviations used in this paper are defined below.

ACM	Application Container and Microservices
API	Application Interface
CMP	Container Management Platform
CSP	Cloud Service Provider
IaM	Identity and Access Management
KMS	Key Management System
OS	Operating System

Appendix B–Glossary	
Application container [NIST SP800-180]	An application container is a construct designed to package and run an application or its components running on a shared operating system. Application containers are isolated from other application containers and share the resources of the underlying operating system, allowing for efficient restart, scale-up, or scale-out of applications across clouds. Application containers typically contain microservices
Container management platform	A container management platform is an application designed to manage containers and their various operations, including but not limited to deployment, configuration, scheduling, and destruction.
Container lifecycle events	The main events in the life cycle of a container are create container, run container, pause container, unpause container, start container, stop container, restart container, kill container, destroy container.
Container rehosting	Redeploying containers on another platform.
Developer [NIST SP 800-64 Rev 2/ ISO/IEC 17789:2014]	The cloud service Developer is a **sub-role** of **cloud service partner** which is responsible for designing, developing, testing and maintaining the implementation of a **cloud service**. This can involve composing the service implementation from existing service implementations. The cloud service Developer's **cloud computing activities** include: • design, create and maintain service components (clause 8.4.2.1); • compose services (clause 8.4.2.2); • test services (clause 8.4.2.3). NOTE 1 – Cloud service integrator and cloud service component Developer describe **sub-roles** of cloud service Developer, where the cloud service integrator deals with the composition of a service from other services, and where cloud service component Developer deals with the design, creation, testing and maintenance of individual service components. NOTE 2 – This includes service implementations and service components that involve interactions with **peer cloud service providers**
Host	OS supporting the container environment
Lateral movement [LIGHTCYBER 2016]	Lateral action from a compromised internal host to strengthen the attacker foothold inside the organizational network, to control additional machines, and to eventually control strategic assets.

Microservices [NIST SP800-180]	A microservice is a basic element that results from the architectural decomposition of an application's components into loosely coupled patterns consisting of self-contained services that communicate with each other using a standard communications protocol and a set of well-defined APIs, independent of any vendor, product, or technology. Microservices are built around capabilities as opposed to services, build on SOA, and are implemented using Agile techniques. Microservices are typically deployed inside application containers.
Microservices architecture	A microservices architecture usually refers to an application that has been structured to use basic elements called microservices, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.
Microservices systems software development	The process of breaking down an application into components (microservices) via code extraction or rewrite, into a microservices architecture of self-contained services that achieve a business objective.
Enterprise Operator	The individual or organization responsible for the set of processes to deploy and manage IT services. They ensure the smooth functioning of the infrastructure and operational environments that support application deployment to internal and external customers, including the network infrastructure, server and device management, computer operations, IT infrastructure library (ITIL) management, and help desk services. [1]
Enterprise Architect	The individual or organization responsible for strategic design recommendations. They determine, by applying their knowledge of cloud, container and microservices components to the problems of the business; the best architecture to meet the strategic needs of the business. Additionally, they develop and maintain solution roadmaps and oversee their adoption working with Developers and Operators to ensure an efficient and effective solution implementation.
Container resources	Four resources required for containers to operate are CPU, memory (+swap), disk (space + speed), and Network
Container resource requests	The amount of CPU, memory (+swap), and disk (space + speed) that the system will allocate to the container considering the resource limit.
Container resource limit	The maximum amount of resources (CPU, memory (+swap) and disk (space + speed)) that the system will allow a container to use.

1 Wikipedia. Information technology operations. Retrieved from https://en.wikipedia.org/wiki/Information_technology_operations.

Appendix C–References

[NIST SP 800-180]	NIST Special Publication (SP) 800-180 (Draft), *NIST Definition of Microservices, Application Containers and System Virtual Machines,* National Institute of Standards and Technology, Gaithersburg, Maryland, February 2016, 12pp. http://csrc.nist.gov/publications/drafts/800-180/sp800-180_draft.pdf
[NIST SP 800-160]	NIST Special Publication (SP) 800-160, *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems,* National Institute of Standards and Technology, Gaithersburg, Maryland, November 2016, 257pp. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v1.pdf
[NIST SP 800-123]	NIST Special Publication (SP) 800-123, *Guide to General Server Security*, National Institute of Standards and Technology, Gaithersburg, Maryland, July 2008, 53pp. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-123.pdf
[NIST SP 800-64 Rev 2]	NIST Special Publication (SP) 800-64 Rev 2, *Security Considerations in the System Development Life Cycle*, National Institute of Standards and Technology, Gaithersburg, Maryland, October 2008, 67pp. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-64r2.pdf
[ISO/IEC 17789:2014]	International Organization for Standardization/International Electrotechnical Commission, *Information Technology – Cloud Computing – Reference Architecture*, ISO/IEC 17789:2014, 2014. https://www.iso.org/standard/60545.html [accessed 5/11/17].
[LIGHTCYBER 2016]	*Cyber Weapons Report 2016,* LightCyber, Ramat Gan, Israel, 2016, 14pp. http://lightcyber.com/cyber-weapons-report-network-traffic-analytics-reveals-attacker-tools/ [accessed 5/11/17].

