The Secure Container Manager Pattern

MADIHA H. SYED, Florida Atlantic University
EDUARDO B. FERNANDEZ, Florida Atlantic University

Software containers have become very popular recently as flexible and portable operating-system-level virtualization solutions. They offer isolated virtual execution environments for applications while sharing host operating system, binaries and libraries with other containers. Containers are used to test and deploy applications. Large numbers of containers in a cluster can be organized, orchestrated and managed by container managers. Container managers facilitate optimal resource utilization for efficient and balanced execution of workloads. Container managers allow automation of tasks related to organization, scheduling and distribution of applications running in containers and they also improve the performance of applications running in them by supporting scalability and replication in distributed application environments. While these systems have a lot of benefits, they can only be utilized to their full potential if they are secure. Container managers incorporate various security features. We present n this paper a pattern for security of container managers.

Categories and Subject Descriptors: **D.2.11 [Software Engineering]**: Software Architectures - Patterns

General Terms: Design

Additional Key Words and Phrases: software containers, architecture patterns, security, virtualization, container cluster management, container ecosystem, security patterns, clouds, cloud ecosystem, microservices.

ACM Reference Format:

Syed, M.H. and Fernandez, E.B. 2018. The Container Manager Security Pattern. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. PLoP (October 2018), 6 pages.

1. INTRODUCTION

Software containers have become popular and convenient virtualization solutions for the cloud computing platform by offering less overhead and more portability as compared to Virtual Machines (VM). Containers offer operating-system-level virtualization where applications are executed in a multitenant environment. Multiple containers share a host operating system (OS), binaries, and libraries, while executing in isolated environments.

Containers may be used to deploy a large number of applications and services in a distributed cluster of hosts. Additional components are required to create, organize, manage, and maintain the container cluster infrastructure which constitute the container manager. They automate the infrastructure management tasks which can be very complex and time consuming if attempted in a non-automatic way and would require a lot more human effort. Container managers configure the environment automatically once a desired state has been specified. Container managers organize containers into groups and improve resource utilization by performing load balancing, auto-scaling and auto-healing.

Containers and container managers have contributed significantly to the popularity and adoption of DevOps. It was found in a study that the container-based approach outperforms the VM-based approach when it comes to performance of DevOps tasks (Kang et al. 2016). In addition, containers also support Microservices architecture (MSA), where an application is defined as a collection of small independently deployable services instead of a single entity. Each service is treated as an application and is executed in isolation in a separate container. Services then use APIs to communicate with other services across containers (Fowler and Levis 2014).

Containers, container manager and their supporting components together constitute a container ecosystem. An ecosystem is a collection of software systems, which are developed and co-evolve in the same environment. We have developed a container reference architecture (RA) as a part of our work on architectural modeling of complex software ecosystems such as cloud, IoT and containers (Syed and Fernandez 2018). An RA is an abstract software architecture which is not tied to the implementation aspects of its domains. These models help understand such complex systems and thereby enable a better utilization of the ecosystems. An RA presents a holistic and unified picture of the system, its components, and their mutual interactions which can simplify

Author's address: Madiha H. Syed, email: msyed2014@fau.edu; Eduardo B. Fernandez, email: ed@cse.fau.edu; Dept. of Computer and Elect. Eng. and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA.

security analysis, implementation of best security practices and evaluation in these systems. We used architectural modeling and patterns to represent cloud ecosystems in our previous work (Fernandez et al. 2015a; 2015b). In addition, we have written various patterns related to virtualization solutions, a Virtual Machine Environment (VME) (Syed and Fernandez 2016) and Software Containers (Syed and Fernandez 2015). Later, we added a pattern for a Container Manager (Syed and Fernandez 2017). A Secure Container pattern described threats and security mechanisms for Software Containers (Syed et al. 2017).

All these patterns can be represented as parts of cloud ecosystem in a pattern diagram (Figure 1) (Fernandez et al. 2016). The Cloud Reference Architecture (RA) is the hub pattern in this ecosystem (Fernandez 2013). Cloud Security RA (SRA) was then built by adding security patterns to the Cloud RA (Fernandez et al. 2015a). The container started as a single component in this Cloud RA model as part of the Platform as a Service (PaaS). Now we have a container ecosystem formed by the container itself, container environment, manager and other related components. More detail on the cloud ecosystem components can be found in earlier work by Fernandez et al. (2016b).

This paper extends the cloud and container ecosystem models and presents a pattern for the Secure Container Manager. Our target audience includes container system administrators, security engineers, system architects, system designers, and software developers who are interested in building container solutions.

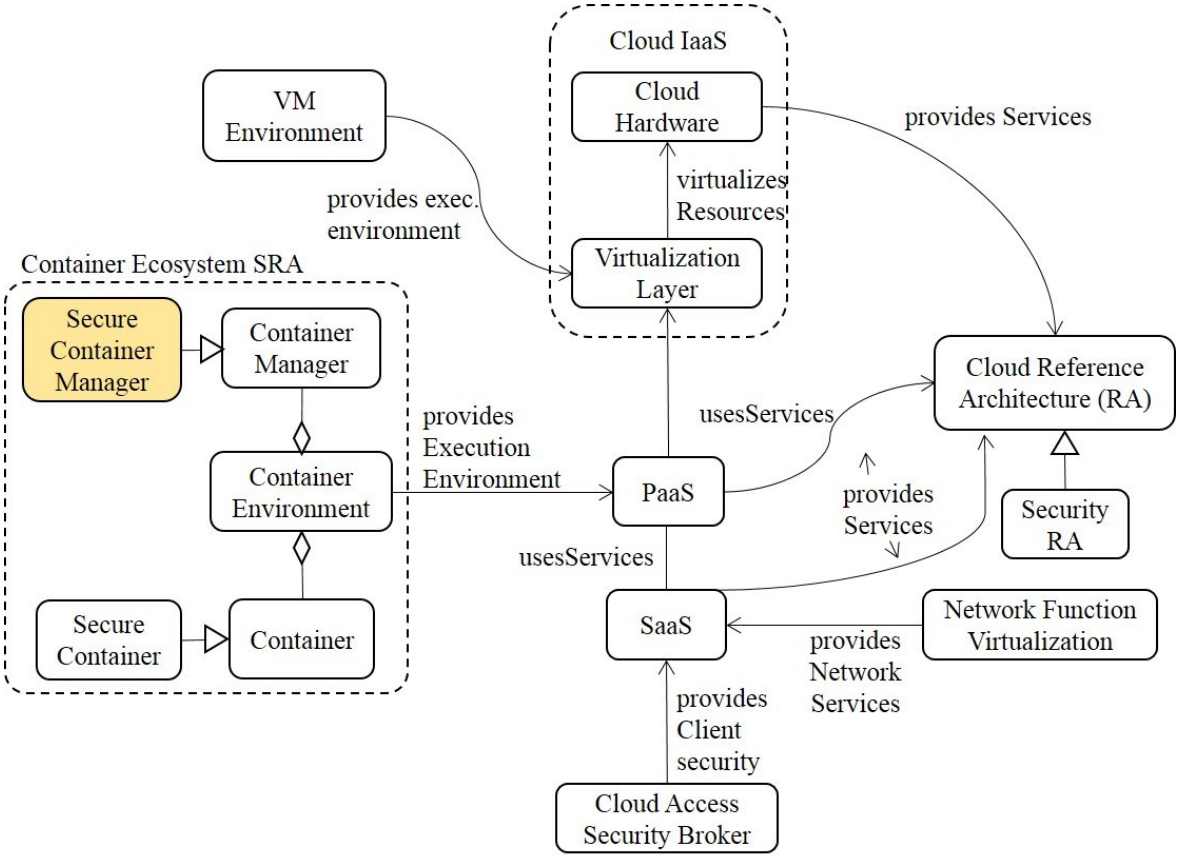

Fig.1. A cloud ecosystem

2. SECURE CONTAINER MANAGER PATTERN

2.1 Intent

A container manager provides means to organize, orchestrate and manage clusters of containers across distributed hosts while providing security by incorporating security countermeasures. These security mechanisms include access control, authentication, logging and encrypted communication.

2.2 Context

Container clusters are used in cloud based environments to execute distributed applications. These container clusters can distribute a large number of containers across multiple hosts. The resulting environment is multitenant and requires communication between containers in the same or across hosts. In addition, container managers traditionally have extensive privileges and environment-wide control over execution of containers. These containers can be hosting variety of applications, with different sensitivities and developed by different teams.

2.3 Problem

Container managers are required to manage and orchestrate containers in a cluster but how do you provide security in this distributed environment? The solution to this problem is guided by the following **forces**:

- *Malicious use of administrative access:* container managers were designed keeping in mind that only administrators will be able to access and interact with them. Administrative access over all the containers running in the cluster was granted with the assumption that the user interacting with the manager will be trustworthy. However, often the container cluster environment may host diverse applications, with varying sensitivity and each managed by different teams. This can raise security concerns if malicious or careless users get access to the system.
- *Unauthorized access to the container manager:* Container managers usually have a separately managed authentication directory to control access to the orchestrator, state storage, etc. Weaker account management of this directory can compromise security of the whole system as the container managers have highly privileged accounts with systemwide access over the container clusters.
- *Data protection:* Containers typically do not use host specific data storage volumes that are managed by the container manager. Application can run in any container across cluster and the data required by it has to be available regardless on where it is stored in the cluster. This can lead to unauthorized access unless data is encrypted at rest to prevent attack.
- *Interference across containers through poorly separated network traffic:* Traffic between various components in containers environment is routed between hosts using overlay networks (Software Defined Networks) which are managed by the orchestrator. If this traffic is encrypted, this means traditional network security tools would not get complete visibility into the traffic to detect and prevent attacks. This traffic can be from different applications and can also be from compromised containers. Compromised containers can access and interfere with other containers as organizations are unable to monitor traffic with their own networks. Attackers may be able to compromise applications with less security (e.g. public facing website) and use them to access shared virtual network to target security sensitive applications.
- *Taking advantage of weak container manager configurations:* Orchestrator node is the most trusted and fundamental node in the cluster. Weak manager configurations can put not only the orchestrator but whole cluster as risk. It can lead to various attacks like unauthorized host becoming part of the cluster, compromise of a single host can jeopardize security of entire cluster (for example if authentication credentials are shared across hosts). Another example of weak configurations is not authenticating or encrypting communication between manager and cluster users/administrators, this can result in eavesdropping [Souppaya et al. 2017].

2.4 Solution

Add security patterns to counteract security threats. Container managers rely on authentication, authorization, access control, secure communication channels, logging and auditing, data encryption in state storage, secure communication, controlling workload/application or user capabilities using security policies, network policies, etc.

Structure:

Figure 2 shows the class diagram for the Secure Container Manager pattern. It is built upon the model of the Container Manager pattern [Syed and Fernandez 2017] and we have added the security patterns to the class diagram. A **Container Manager** orchestrates and manages the **Cluster** of **Containers** which are executed across multiple **Hosts**. Container manager organizes the containers by grouping closely-related containers into **Coherent Units (CUs).** The CU is treated as a single entity and is assigned an IP address which only changes when its host changes. The **Resource Monitor** gathers and stores data for each host such as its health, resource utilization, etc. Using the data provided by the resource monitor, the **Scheduler** assigns CUs to hosts. The CUs

communicate with each other via the **Overlay Network.** A **Secure Channel** for cluster-wide communication and network traffic is obtained using encryption. The services/applications in a cluster use the **Discovery Service** to look each other up. Cluster state information is stored in the **State Storage** which can use **Encryptor** to protect the data at rest. **Authenticator** and **Authorizer** are used to limit and control capabilities of users/containers interacting with the container manager. A **Security Logger/Auditor** is used to keep track of all security activities in the container cluster.

There are other security measures related to container image protection and individual software container security have been addressed in our previous work [Syed et al. 2017]. We focus here only on security aspects of container managers.

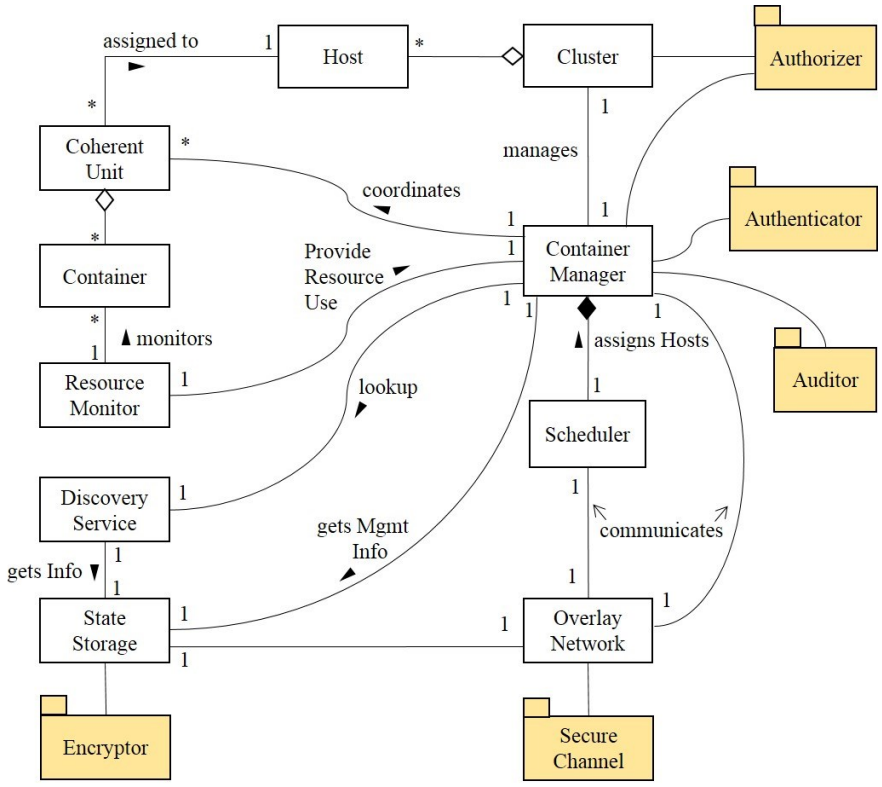

Fig.2. Secure Container Manager Class diagram

2.5 Implementation

As shown in figure 2, most of the components in a secure container manager can be described by patterns which can be applied when implementing these systems. For example, Role Based Access Control (RBAC) is used as an authorizer to control access to CUs.

Container managers offer various security features as we have described in the previous section. Not all container managers offer all of these features. Figure below is from techtarget report shows container management features provided by various platform.

Container Management Technology	**Purpose**	**Examples of Software Products and Vendors**
Scheduling and Orchestration	Scheduling and orchestration simplifies container administration for densely populated hosting environments.	Kubernetes, Mesos, red Hat OpenShift, Apprenda Kismatic Enterprise Toolkit, Google Container Engine (GKE), Ubuntu, Canonical Distribution of Kubernetes, Rancher, Univa Navops, VMware Photon Platform
Security	Container secrets management tools protect passwords and tokens in secure environments. Other security tools inspect container images and track behavior on the network.	Kubernetes, Mesosphere, CISOfy Lynis, HashiCorp Vault, Docker Notary, Aqua, Twistlock, NeuVector, StackRox, Deepfence, Tenable, Trend Micro Deep Security

Storage	Container Storage utilities handle the transfer of container image files from registries to storage systems assigned to apps in production.	Rex-Ray, Docker Infinit, Portworx, Blockbridge Networks, Red Hat Container-Native Storage
Networking	Virtual networking supports container deployments, which scale and change rapidly.	Contiv, CoreOS, Weaveworks, Project Calico, Ansible Container, VMware NSX, Cisco Application Centric Infrastructure
Monitoring	Specialized monitoring tools track performance, bugs and security in containerized workloads	Sysdig, Docker Stats, Google cAdvisor, Prometheus for Kubernetes, New Relic, Datadog, Cisco AppDynamics, IBM, Dynatrace, SignalFx

Fig.3. Container management tools and their implemented features [Techtarget 2018]

Each of these features can strengthen the container cluster security but they need to be configured properly. More general guidelines and recommendations about security configurations can be found in NIST application container security guide [Souppaya et al. 2017]. Specific configurations may vary based on container manager implementations.

Kubernetes, for example, integrates a component for RBAC that cross checks incoming user or group to set permissions based on their role. Node and RBAC authorizers can be used in addition to NodeRestriction admission plugin. It limits the host and CUs that can be modified by a kubelet(Container manager local agent on the host). When kubelet uses system:nodes group credentials and username in system:node:<nodeName> form, it will only be allowed to modify their own Node(Host) and Pod (CU) API object (bound to their host) [Kubernetes 2018]. In order to prevent the malicious use of administrator resources, Kubernetes engine has by default disabled the Kubernetes Dashboard, which was the web user interface that used very privileged service account. Kubernetes engine has to explicitly grant permissions for allowing the storage access and manipulation of compute -rw resources through node's service account. New clusters do not get compute -rw and storage-ro scopes by default. It can however be allowed through configurations. The detailed examples on how to make these configurations and create and grant permissions using RBAC in Kubernetes can be found in [Small 2018].

2.6 Known Uses

Container manager frameworks like Google Kubernetes [Kubernetes 2017], Docker Swarm [Swarm 2017] and Marathon [Mesosphere 2017] include various security features in the platform.

2.7 Consequences

- *Malicious use of administrative access:* All access to the container managers should require authentication and access control. RBAC can be used to only grant access when absolutely required. Only allow users to perform the specific actions that are required by their role on the specific cluster components, rather than having unrestrained administrative control over whole cluster. Least privilege access model should be used for container managers.
- *Unauthorized access to the container manager:* Authentication and security policies can be used to control access to the container manager. Administrative accounts can be required to use strong authentication methods, e.g. multifactor authentication, single sign-on, strong credentials, etc.
- *Data protection:* State storage can be encrypted at rest to avoid chances of unauthorized access; however, it can add overhead.
- *Interference across containers through poorly separated network traffic:* Container communications takes place on a secure channel to prevent interference. In addition, the network can be configured to use separate virtual networks for applications with different levels of sensitivity. Containers can be grouped based on security sensitivity. This can be achieved by applying security policies through names and labels. Deployments can also be restricted to specific hosts based on security requirements. Besides encryption of the traffic between containers, network connections can also be configured to require mutually authenticated. Container environment specific filtering tools can be integrated in the cluster to add security [Souppaya et al. 2017].
- *Taking advantage of weak container manager configurations:* Configurations can be specified in the manager to authenticate a host before addition to the cluster, maintain host identity throughout lifecycle, and keep

an accurate record of hosts and their connectivity states. limit application/container capabilities and encrypting communication between manager and cluster users/administrators.

2.8 See also (related patterns)

- Software Container [Syed and Fernandez 2015]. A Software Container provides an execution environment for applications sharing a host operating system, binaries, and libraries with other containers. Containers have less execution overhead but are less flexible and secure than VMs.
- Secure Software Container [Syed et al. 2017] – describes security threats and their countermeasures to define software containers.
- Network Functions Virtualization (NFV) (Fernandez and Hamid 2015). An architecture for the construction of network services using software building blocks. The building blocks, Virtual Network Functions (VNFs), are typically created from cloud services using virtual machines or containers.
- Virtual Machine Operating System [Fernandez 2013] --provides a set of replicas of the hardware architecture (Virtual Machines) that can be used to execute (maybe different) operating systems with a strong isolation between them.
- Container Manager Pattern [Syed and Fernandez 2017] acts as a high-level controller to orchestrate and manage clusters of containers across distributed hosts. It provides functions for organizing, deploying, managing, and securing container clusters.

REFERENCES

Docker. 2017. Docker stats. Retrieved April 16, 2017 from https://docs.docker.com/engine/reference/commandline/stats/

Fernandez, E. B. 2013. *Security patterns in practice: Designing Secure Architectures Using Software Patterns*. J. Wiley & Sons, Inc.

Fernandez, E. B., Monge, R. and Hashizume, K. 2015a. Building a security reference architecture for cloud systems. *J. Requirements Engineering* (June 2015), 1–25. DOI:10.1007/s00766-014-0218-7

Fernandez, E.B., Yoshioka, N. 2011. Two patterns for distributed systems: Enterprise Service Bus (ESB) and Distributed Publish/Subscribe. In *Proceedings of the 18th Conference on Pattern Languages of Programs* (PLoP '11). ACM, New York, NY, USA, Article 8, 10 pages. DOI=http://dx.doi.org/10.1145/2578903.2579146

Fernandez, E. B., Yoshioka, N. and Washizaki, H. 2015b. Patterns for Security and Privacy in Cloud Ecosystems. In *2nd International Workshop on Evolving Security and Privacy Requirements Engineering* (ESPRE 2015), IEEE (August 2015), 13-18. DOI:10.1109/ESPRE.2015.7330162

Fernandez, E. B., Yoshioka, N., Washizaki, H. and Syed, M. H. 2016. Modeling and security in cloud ecosystems. *Future Internet* 2016, 8(2), 13; doi:10.3390/fi8020013 (Special Issue Security in Cloud Computing and Big Data)

Fowler, M. and Levis, J. 2014. Microservices. Retrieved April 15, 2017 from https://martinfowler.com/articles/microservices.html

Kubernetes. 2017. Production-Grade Container Orchestration. Retrieved February 2. 2017 from https://kubernetes.io/

Kubernetes. 2018. Securing a Cluster. https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/

Mesosphere. 2017. Meet Marathon: Production-ready container orchestration at scale. Retrieved February 2. 2017 from https://mesosphere.com/blog/2016/02/17/marathon-production-ready-containers/

Small, Aaron. 2018. Exploring container security: Running a tight ship with Kubernetes engine 1.10. Retrieved June 2. 2018 from https://cloudplatform.googleblog.com/2018/04/Exploring-container-security-Running-a-tight-ship-with-Kubernetes-Engine-1-10.html

Souppaya, M., Morello, J. and Scarfone, K. 2017. Application Container Security Guide. Retrieved September 25. 2017 from https://doi.org/10.6028/NIST.SP.800-190

Swarm. 2017. Docker Swarm. Retrieved February 2. 2017 from https://www.docker.com/products/docker-swarm

Syed, M. H. and Fernandez, E. B. 2015. The Software Container pattern. In *Proceedings of the 22nd Conference on Pattern Languages of Programs* (PLoP 2015). Pittsburgh, PA, October 24-26, 2015

Syed, M. H. and Fernandez, E. B. 2016. A pattern for a virtual machine environment. In *Proceedings of the 23rd Conference on Pattern Languages of Programs* (PLoP 2016), Monticello, IL, October 24-26, 2016

Syed, M. H. and Fernandez, E. B. 2017. The Container Manager Pattern, 22nd European Conference on Pattern Languages of Programs (EuroPLoP 2017), Germany, July 12-16, 2017

Syed, M. H., Fernandez, E. B. and Silva P. 2017. The Secure Software Container Pattern, 23rd Conference on Pattern Languages of Programs (PLoP 2017), Vancouver, Canada, October 22-25, 2017

Syed, M. H. and Fernandez, E. B. 2018. A reference architecture for the container ecosystem. ARES 2018, Germany, accepted for publication.

Techtarget. 2018. Container Management Software. Retrieved February 2. 2018 from https://searchitoperations.techtarget.com/definition/container-management-software

Kang H., Le M. and Tao S. 2016. Container and Microservice Driven Design for Cloud Infrastructure DevOps. 2016 IEEE International Conference on Cloud Engineering (IC2E), Berlin, 2016, pp. 202-211.doi: 10.1109/IC2E.2016.26