Security
of Kubernetes Containers –
- Holistic View

Tomasz Kruk
Faculty of Electronics and Information Technology
Warsaw University of Technology

www.linkedin.com/in/tomasz-jordan-kruk
tomasz.kruk@pw.edu.pl

List of Content

1. Dependability
2. Kubernetes – main goals
3. Security: assets, threat actors and controls
4. 4C - cloud native security
5. Kubernetes in CVE database

Dependability of Computer Systems

Dependability

(*strict definition*)	The ability to deliver service that can justifiably be trusted
(*extended definition*)	The ability to avoid service failures that are more frequent and more severe than is acceptable

Dependable computer system

The computer system representing the dependability feature. The computer system one may depend on/rely on

Attributes of dependability

reliability	continuity of correct service
availability	readiness for correct service
maintainability	ability to undergo modifications and repairs
safety	absence of catastrophic consequences on the user(s) and the environment

Attributes of security

confidentiality	the absence of unauthorized disclosure of information
integrity	absence of improper system alterations
availability	readiness for correct service (*as above*)

Kubernetes – what is it for?

Kubernetes provides you with a framework to run distributed systems resiliently.

- **scalability** handling
- application **failover** handling
- **deployment patterns** providing

For example: can easily manage a canary deployment

Kubernetes – what does it provide?

What Kubernetes provides the user with

- service discovery and load balancing
- storage orchestration
- automated rollouts and rollbacks
- automatic bin packing
- self-healing
- secret and configuration management

Kubernetes – what should it be completed with?

- Kubernetes is not a traditional, all-inclusive PaaS (Platform as a Service) system.

- Kubernetes operates at the container level rather than at the hardware level, it provides some generally applicable features common to PaaS offerings, such as
  - deployment
  - scaling
  - load balancing

  and lets users integrate their
  - **logging**
  - **monitoring** and
  - **alerting** solutions

Web architecture, comparison: **assets**

Traditional web app

- web server
- application server
- database server
- hosts

Web app on K8s

- web server
- application server
- database server
- nodes (worker + master)
- pods
- persistent volumes
- K8s components (api-server, etcd, proxy, kubelet, scheduler, cntrllr-manager)

Web architecture, comparison: threat actors

Traditional web app

- Internet/end users
- internal attackers
- admins

Web app on K8s

- Internet/end users
- internal attackers
- admins
- malicious/compromised nodes
- malicious/compromised pods
- compromised K8s components
- apps running inside the cluster

Web architecture, comparison: security controls

Traditional web app

- firewall
- DMZ
- Internal network
- WAF
- TLS connections
- file encryption
- database authorization
- database encryption

Web app on K8s

- network policies
- TLS, mTLS
- pod security policy
- WAF
- pod isolation
- file encryption
- database authorization
- database encryption
- admission controllers
- K8s authorization

Web architecture
comparison **summary**

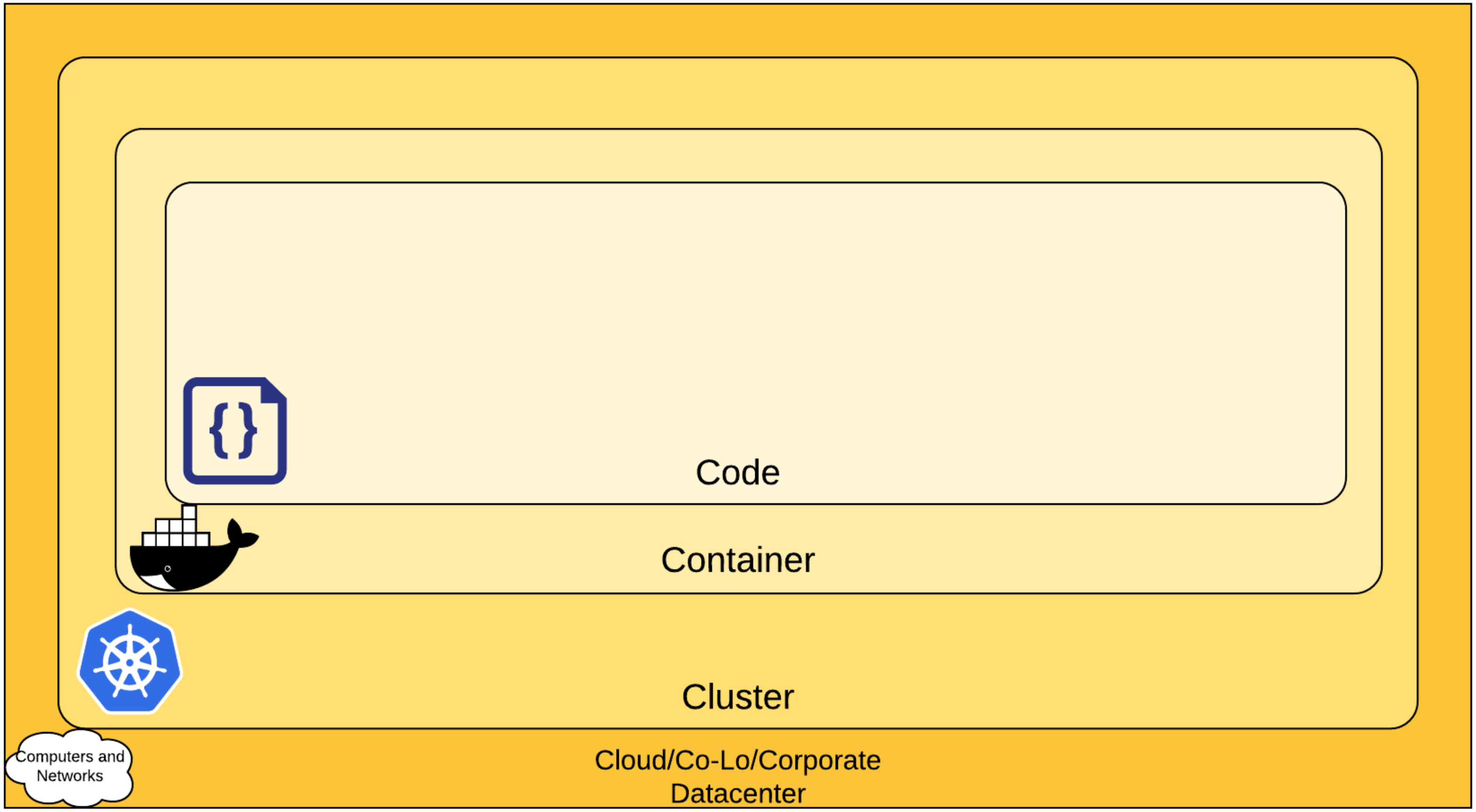
- more assets to be protected in a cloud-native architecture,
- more threat actors in this space,
- Kubernetes provides more security controls, but also more complexity.

Complexity is the enemy of security.

Necessary to do both:

- application threat modeling, and
- infrastructure threat modeling - together.

The four `C` letters of Cloud Native Security



(from Kubernetes doc)

First `C`: Cloud Infrastructure

1. Area of concern for cloud infrastructure:

- network access to API Server (control plane)

  not allowed publicly, controlled by NAC lists restricted to IPs required for cluster administration

- network access to nodes

  only accept connections from the control plane on the specified ports, and NodePort and LoadBalancer services; if possible, nodes unreachable for the public at all

- Kubernetes access to Cloud Provider API

  following principle of least privilege

- access to etcd

  limited to the control plane only, use etcd over TLS if possible

- etcd encryption

  keep etcd (containing Secrets) encrypted at rest

Second `C`: Cluster

2. Area of concern for workload security: (https://kubernetes.io/docs/ ↵)

- RBAC authorization (access to the Kubernetes API)
  reference/access-authn-authz/rbac/
- authentication
  concepts/security/controlling-access/
- application secrets management (and encrypting in etcd at rest)
  concepts/configuration/secret/ tasks/administer-cluster/encrypt-data/
- pod security policies
  policy/pod-security-policy/
- quality of service (and cluster resource management)
  tasks/configure-pod-container/quality-service-pod/
- network policies
  concepts/services-networking/network-policies/
- TLS for Kubernetes ingress
  concepts/services-networking/ingress/#tls

Third `C`: Container

3. Area of concern for containers:

- container **vulnerability scanning** and **OS dependency** security

  as part of an image build step, one should scan his/her containers for known vulnerabilities.

- **image signing** and enforcement

  one should sign container images to maintain a system of trust for the content of the containers

- **disallow privileged** users

  while constructing containers, one should only create such users inside of the containers that have the least level of necessary operating system privilege

Fourth `C`: Code

4. Area of concern for code:

- access over TLS only

  encrypt network traffic between services with mTLS - a two sided verification of communication between two certificate holding services.

- limiting port ranges of communication

  only expose the ports on a service essential for communication or for metric gathering

- 3rd party dependency security

  regularly scan application's third party libraries for known security vulnerabilities

- static code analysis

  perform checks using automated tooling that can scan codebases for common security errors

  https://owasp.org/www-community/Source_Code_Analysis_Tools

- dynamic probing attacks

  run automated tools against your service to try some of the well-known service attacks (including SQL injection, CSRF, and XSS)

  for example: OWASP Zed Attack proxy

Kubernetes security vulnerabilities

Publicly known security vulnerabilities of Kubernetes.

1. CVE-2019-11246 - a **path-traversal** issue allowed attackers to modify the content on the client side, which could potentially lead to exfiltration or code execution on the cluster administrator's machine.
2. CVE-2019-1002100 - allowed users to cause **Denial-of-Service** (DoS) attacks on the API server.
3. CVE-2019-11253 – **improper input validation** allowed unauthenticated users to cause DoS attacks on kube-apiserver.
4. CVE-2019-11247- allowed users with **namespace privileges** to modify cluster-wide resources.

Upgrading to the latest version of Kubernetes and kubectl, which patches vulnerabilities, should be on the daily operations priority list.