Secure Modern Application Delivery with SUSE® CaaS Platform

Roger Klorese
Senior Product Manager
roger.klorese@suse.com
Agenda

Container Security Concerns

Addressing Container Security

Security @ SUSE
Container Security Concerns
True or False?

Containers are inherently insecure.
Some Learnings from an Enterprise Study

94%: “Containers have security implications”
31%: “Worried about the lack of mature security solutions for containers”
31%: “Current server security solutions do not support containers”
28%: “A single infected container could easily spread to others”
16%: “Portability of containers means they could be more susceptible to ‘in motion’ compromise”
Addressing Container Security
Security Requirements

- Enforcing the deployment of a secure gold image on container hosts, using governance and policies.
- Role-based access control to the platform itself and the containers.
- Runtime and at-rest scanning.
- Network segmentation and access control.
- Network visibility.
- Encryption in motion.
- Secret management, to avoid having secrets such as database passwords in container images.
- Runtime security.
- Monitoring the security posture of the platform, using classical security tools.
Secure Gold Image

Enforcing the deployment of a secure gold image on container hosts, using governance and policies

Best Practice:

• Build gold master container image based on SLES base containers
• Integrate CI/CD pipeline to deliver applications and app updates consistently and securely
Role-Based Access Control

Role-based access control to the platform itself and the containers

Best Practice:
(In decreasing order of security)
• Create service account for application with only the permissions it needs
• Create service account for application that has admin access to the application’s namespace
• Grant admin access to the default service account for a particular namespace to that same application namespace

WORST Practice:
Disable RBAC, or grant all permissions on workloads to kube-system
Scanning

Runtime and at-rest scanning.

Best Practice:
• Build containers with methodology that performs at-rest scanning
  • SUSE Manager
  • Third-party scanners integrated into CI/CD pipeline
    • Jfrog, Aqua – also perform runtime scanning
Network Policies

Network segmentation and access control.

Best Practice:
• Leverage Cilium in SUSE CaaS Platform 4 to:
  • Control ingress and egress to the cluster
  • Control ingress and egress to namespaces
• Consider SUSE CaaS Platform Ready partner products such as container firewalls

```bash
cat <<EOF | kubectl create -n empire -f -
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "allow-within-namespace"
specs:
  - endpointSelector:
    matchLabels: {}
    egress:
    - toEndpoints:
      - matchLabels:
        "k8s.io.kubernetes.pod.namespace": empire
      ingress:
      - fromEndpoints:
        - matchLabels:
          "k8s.io.kubernetes.pod.namespace": empire
  - endpointSelector:
    matchLabels: {}
    egress:
    - toEndpoints:
      - matchLabels:
        "k8s.io.kubernetes.pod.namespace": kube-system
        "k8s.io.k8s-app": kube-dns
    ingress:
    - fromEndpoints:
      - matchLabels:
        "cilium.io":
EOF
```
Visibility

Network visibility.

Best Practice:
• to monitor network traffic, security, and performance:
  • Deploy Prometheus (from upstream) with SUSE CaaS Platform 3
  • Deploy Prometheus delivered with SUSE CaaS Platform 4
  • Consider SUSE CaaS Platform Ready Partner products
Encryption in Motion

Best Practice:

- Utilize the in-motion encryption encryption within the cluster delivered by default with cluster-signed certificates

- Add customer-supplied trusted-root certificates for external interfaces (API-server, Dex directory services, etc.)
Secret Management

Secret management, to avoid having secrets such as database passwords in container images.

Best Practice:

- Access secrets from environment variables
- If you use mounted secrets, enable encryption at rest (not yet “stable”/released)
- Consider third-party secrets storage solutions
Runtime Security

Best Practice:
• Use Pod Security Policies (PSPs) to control:
  • Use of privileged containers
  • Use of host resources (file systems, networks, etc.)
  • Privilege escalation
  • Linux capabilities
  • OS security profiles
• Consider use of partner products for runtime security monitoring
Platform Security

Monitoring the security posture of the platform, using classical security tools.

Don’t forget there is a platform underneath the container environment!

Best Practice:

- OS-level security tools and profiles
- Physical and virtual network security tools:
  - Firewalls, WAF, IPS, anti-malware
- Storage and cloud security policies
Governance Examples

- Containers cannot be started by a user using a shell on the host or by the remote Docker CLI.
- A set of workloads should run on the same hosts (affinity) or cannot run on the same host (anti-affinity).
- Kubernetes deployment can only be created using Helm.
- Transmission between nodes should be encrypted.
- Data at rest should be encrypted.
- Secrets should be centrally managed and encrypted.
- Only specific groups of users can start and stop containers belonging to a particular application (RBAC applied to scheduling).
- Certain apps need a dedicated namespace.
- YAML files must be managed subject to revision control and RBAC.
“The Low-Hanging Fruit”

- Disable anonymous access
- Disable automounting the default service account token
- Use admission control to block privilege escalation by shell access on privileged containers
- Limit user impersonation
- Disallow privileged containers – or if needed, control individual privileges
- Disallow or restrict sharing of host PID namespace, IPC namespace, and network stack
- Use resource limits to mitigate “noisy neighbor syndrome”
- Patch promptly!
- TRAIN DEVS AND DEVOPS IN SECURITY CONSIDERATIONS!
Security @ SUSE
Security @ SUSE

- Engineering security team involved in design and review
- Key security audits run against releases
- SUSE receives early notification of vulnerabilities and remediation
  - General software channels across all components
  - Specifically from the Kubernetes project
- If vulnerable, patches are shipped promptly as maintenance updates
And Finally...
More Containers Content @ SUSECON 19

- Best Practices in Deploying SUSE CaaS Platform [TUT1131]
  - Tuesday @10:15, Wednesday @2:00
- Enabling Business Continuity with SUSE CaaS Platform [BOV1078]
  - Tuesday @2:00
- SUSE CaaS Platform Hands-On [HO1209]
  - Tuesday @4:30, Wednesday @2:00
- Bringing container security to the next level using Kata containers [TUT1201]
  - Tuesday @4:30, Wednesday @3:15
- GitLab on SUSE CaaS Platform [HO1415]
  - Tuesday @10:15, Thursday @2:00
- Integrating Identity with LDAP for SUSE CaaS Platform [TUT1254]
  - Tuesday @10:15, Thursday @3:15
More Security Content @ SUSECON 19

- Automate Security Testing and System Compliance [TUT1220]
  - Thursday @10:00
- Secure by default - anti-exploit techniques and hardenings in SUSE products [TUT1046]
  - Tuesday @10:15, Wednesday @2:00
- Security, Low costs and Excellent Performance [BOV1146]
  - Thursday @10:00
- SUSE Security Roadmap [FUT1210]
  - Tuesday @3:15, Thursday @10:00
- Tymlez Blockchain on SUSE CaaS Platform [BOV1313]
  - Tuesday @10:15