Leveraging the Serverless Architecture for Securing Linux Containers

Nilton Bila, Paolo Dettori, Ali Kanso, Yuji Watanabe*, Alaa Youssef
IBM T.J. Watson Research Center – New York
*IBM Research – Tokyo, IBM Japan Ltd.

Ali Kanso, PhD
Senior Cloud SW Engineer
IBM T.J. Watson, NY
Leveraging the Serverless Architecture for Securing Linux Containers
But Container images can have **vulnerabilities** baked in them!
Software Vulnerabilities

- E.g.: Ghost Vulnerability in glibc library < 2.18 (2000 à 2013)
- E.g.: OpenSSH installed &
  - PasswordAuthentication yes
  - PermitEmptyPasswords Yes (or even no)

Malware Signature
- Viruses
- Worms
- SpyWares
- Trojan Horses
Scanning for Vulnerabilities

- Scan images and deployed containers
- Vulnerabilities in installed software packages
- Security configuration checks
- Malware signature detection

IBM Vulnerability Advisor

Docker Security Scanning
Clustering Containers

Clustering can be overwhelming

Kubernetes can help
What is Kubernetes?

- Applications
- Containerized
- Clustered

**Master Node**
- Kubernetes master
- Scheduling
- Monitoring
- Recovery management
- Auto-scaling
- Authorization/Authentication...

**Worker Node (Minion)**
- Containerized apps
- Kubernetes agent
- Monitoring
- Reporting
- Executing master’s recommendations...

**Kubernetes master**
- Scheduling
- Monitoring
- Reporting
- Executing master’s recommendations...
Example pod description

kind: Pod
metadata:
  name: myPod
spec:
  containers:
    - name: sleep-forever
      image: pause:0.8.0
      resources:
        limits:
          memory: 1000Mi
K8s APIs

monolithic v1 API

REST path /api/v1
✓ Pods
✓ Services
✓ Replication controllers
✓ Resource quotas
✓ Nodes
✓ Endpoints
✓ ...

REST path /apis/extensions/$VERSION
✓ Deployments
✓ HorizontalPodAutoscalers
✓ Ingress
✓ Jobs
✓ DaemonSets
✓ Third party resources
✓ ...

K8s Operators

- K8s API
- Third party resources
- Operators

<<Extend>>

<<Leverage>>
K8s Third Party Resource (TPR)

http://192.168.0.15:8080/apis/myorg.com/v1/namespaces/default/securityactions/quarantine

Controller: bridging the actual state with the desired state

Resource instance: reflecting the desired state
Security action, to quarantine or delete container

Third-Party Software (executable)

<<Watch & react>>

K8s Master (API server)

<<CRUD operations>>

API path
- TPR
- Securityactions
  - quarantine

Controller: bridging the actual state with the desired state
Kubernetes Limitation

- K8s does not implement the needed range of actions to contain a threat
  - Limited to: Kill pod, Rolling-Upgrade (involves killing)

_We need to have severity-based actions!_
Introducing the Security Enforcement Operator

Kubernetes master

- K8s API-server
- Docker
- OS

Kubernetes Worker

- Pods
- SEO
- Docker
- Net-plugin
- OS

- Quarantine/Unqarantine
- Pause/unpause
- Stop/start
- Fast-delete
- Graceful-delete

Based on scanning results
Vulnerability Scanner

Image Registries

Registry Monitor

Vulnerability Scanner

security configuration checks
package vulnerability detection
malware signature detection

Thread Intelligence

ingest threat data periodically

Notification + summary of vulnerability findings

K8s Worker Nodes

Pods

K8s Workers

Image Registries

scan images

Registry Monitor

VS-agent

image configuration information

scan images

Vulnerability Scanner

container configuration information

scan deployed containers

VS-agent

container configuration information

(Report from Vulnerability Advisor)
VS Report Example

• Identify specific software package versions in the container with disclosed vulnerabilities

<table>
<thead>
<tr>
<th>Affected Packages</th>
<th>Security Notice</th>
<th>Description</th>
<th>Corrective Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>eject</td>
<td>3285-1</td>
<td>Eject could be made to run programs as an administrator.</td>
<td>Upgrade eject to at least version 2.1.5+deb8-s390x64-1ubuntu2.04.1</td>
</tr>
<tr>
<td>libcuse-1-3</td>
<td>3215-1</td>
<td>Several security issues were fixed in CDBus.</td>
<td>Upgrade libcuse-1-3 to at least version 1.5.15-6ubuntu4.4</td>
</tr>
<tr>
<td>libgmp10</td>
<td>3289-1</td>
<td>Libgmp incorrectly generated random numbers.</td>
<td>Upgrade libgmp10 to at least version 4.10.1-4ubuntu4.4</td>
</tr>
<tr>
<td>libgmp14</td>
<td>3295-1</td>
<td>Libgmp could be used to expose sensitive information.</td>
<td>Upgrade libgmp14 to at least version 4.10.1-4ubuntu4.4</td>
</tr>
<tr>
<td>tar</td>
<td>3132-1</td>
<td>tar could be made to overwrite files.</td>
<td>Upgrade tar to at least version 1.27-1ubuntu0.1</td>
</tr>
</tbody>
</table>

• Identify specific issues with the container configurations

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
<th>Corrective Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improvement Needed</td>
<td>PASS_MIN_DAYS must be set to 1</td>
<td>Minimum days that must elapse between user-initiated password changes should be 1.</td>
</tr>
<tr>
<td>Improvement Needed</td>
<td>PASS_MAX_DAYS must be set to 90 days</td>
<td>Maximum password age must be set to 90 days.</td>
</tr>
<tr>
<td>Improvement Needed</td>
<td>Minimum password length not specified in /etc/pam.d/common-password</td>
<td>Minimum password length must be 8.</td>
</tr>
<tr>
<td>No Improvement Needed</td>
<td>No found malware</td>
<td>Remove malware from container/image.</td>
</tr>
</tbody>
</table>
Leveraging the Serverless Architecture for Securing Linux Containers
Introducing OpenWhisk

- **K8s Workers** vs **OpenWhisk**
- **K8s API server** vs **-agent**
- **CRUD operations**
- **Add/Remove/Modify**
- **Action based policies**
Why OpenWhisk?

• OpenWhisk is the Glue between VS and K8s, it *enables*:
  – Different policies for different users
  – Multiple Clusters register to the same OpenWhisk deployment
  – Central point of policy management across clusters
Report API and Notifications on Vulnerability Scanner

• Supports scans for **multiple registered Kubernetes** clusters.
• Provide **RESTful APIs** for access to Vulnerability reports for each container
• Use authentication token to **restrict access** to cluster data at the granularity of Kubernetes namespaces.
• **Notify events** with new **vulnerability findings** to registered OpenWhisk API endpoints.
• Trigger **action invocations** to the **OpenWhisk API endpoints** registered for the Kubernetes cluster.
Notifications

• User creates action with known URL endpoint:
  – https://openwhisk.ng.bluemix.net/api/v1/web/<USER>/policy

• Vulnerability Scanner posts vulnerability notification to **policy endpoint**

```json
{
  "clusterid": "xyz",
  "podid": "nginx-3382653011-3p4p0",
  "vulnerability_type": "package",
  "vulnerability_status": "vulnerable"
}
```
• import vs
  import kubernetes

def main(params):

    findings = vs.get_findings(pod_id, timestamp)
    vulnerable_packages = findings['vulnerable_packages']
    insecure_configs = findings['insecure_configurations']

    if len(vulnerable_packages) > 0:
        kubernetes.snapshot(pod_id)
        kubernetes.terminate_graceful(pod_id)
        return {'text': 'Deleted pod ' + pod_id}

    if 'remote_shell_installed' in insecure_configs:
        kubernetes.quarantine(pod_id)
        Terminate_faste(pod_id)
        return {'text': 'Quarantined pod ' + pod_id}

    return {'text': 'Container was not modified ' + pod_id}
Interaction Summary

New Pod created

K8s
- Scanning triggered
  - Vulnerability found
  - Quarantine Pod
  - Threat contained
  - Pod quarantined

VS
- Vulnerability found
  - Quarantine Pod

OpenWhisk
- Execute policy

SEO
- Quarantine container(s)
  - Container(s) quarantined

Networking
**Related Work**

*Starlight* implements a kernel module that intercepts local operations on each host and passes them to a local agent which in turn passes them to an event processor that analyzes the event and determines whether or not to alert the admin.

*LiCShield* generates AppArmor profiles by tracing the container engine (Docker daemon) during the build and the execution of the containers.

**Securing Containers**

**Using Serverless in the Cloud**

*Docker Security Scanning* can scan images in private repositories to verify that they are free from known security vulnerabilities or exposures, and report the results of the scan for each image tag.

**OpenSCAP** (Security Content Automation Protocol) searches for an appropriate fix element, resolves it, prepares the environment, and executes the fix script.

**Lambdefy framework** to demonstrate the differing requirements between applications deployed to IaaS and applications deployed as a cloud event, and Media Management System for showing high scalability of image resizing tasks on Lambda.
That’s it! Questions?

Leveraging the Serverless Architecture for Securing Linux Containers

- OpenWhisk
- Kubernetes
- Vulnerability scanner
- Security Enforcement Operator