BLUEHAT SHANGHAI 2019

Is my Container Secure?

Study of Vulnerability in Container World

Cecilia Hu, Yue Guan, Zhaoyan Xu

Palo Alto Networks

Agenda

- <u>Introduction</u>
- Basic Statistics
- Study of Vulnerability
- Image in your Cluster
- Practical Suggestions

Background

- Containers have recently become a popular approach to provision Micro-service over the Cloud.
- With more advanced cloud applications deployed, the security risks of images becomes a big headache for DevOps team.
- We want to know how bad is the situation and how we could defense against the threats.

Motivation

In this talk, we will cover:

- How is the state-of-art status of container image security?
- How to measure the security of container image in your application environment?
- How to mitigate threats from the vulnerable container images?
- What is best practice for securing your images?

Agenda

- Introduction
- Data Collection
- Study of Vulnerability
- Image in your Cluster
- Practices Suggestions

Approach

We study the problem by:

STEPI: We crawl public-available images repositories from multiple sources, such as DockerHub and Github.

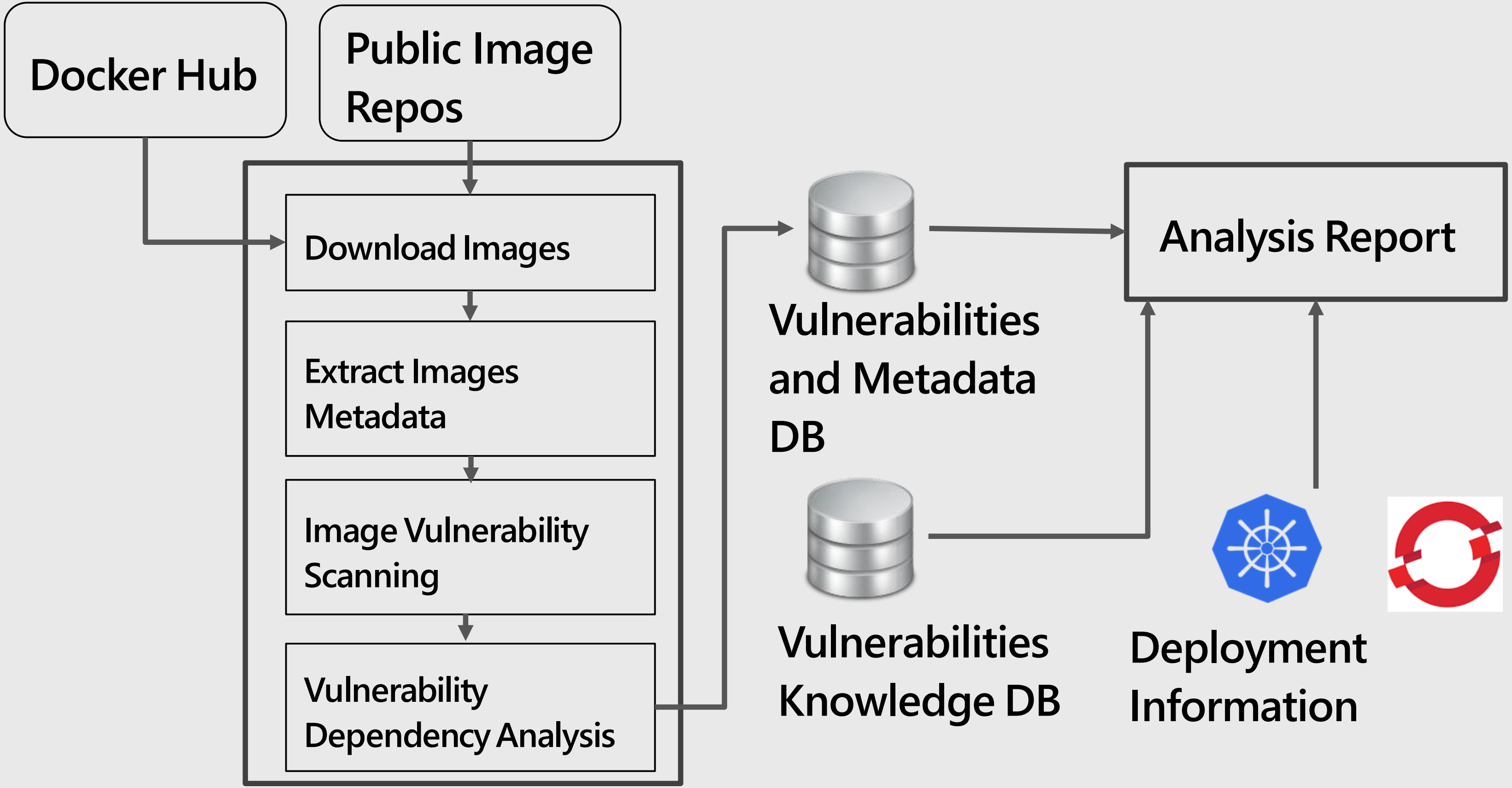
STEP II: We scan these images using container image scanners and get vulnerabilities information for these images.

STEP III: We analyze these vulnerabilities using our VulnerDB and discloses trends for these vulnerabilities.

Data Source

Source	Description
DockerHub	Image Repository where we can directly download/crawl docker images
Clair	Open Source Vulnerability Scanner to get system-wide vulnerabilities information
hub.docker.com	Image scanning result for public assess
Container Analysis API	Image analysis service provided by Google Cloud
Dockerfile from Github	From the pull count, we can get popular open-source projects which could be deployed as container

Data Collection



Docker Hub

Public Image Repos

Download Images

Extract Images Metadata

Image Vulnerability Scanning

Vulnerability Dependency Analysis

Vulnerabilities and Metadata DB

Vulnerabilities Knowledge DB

Analysis Report

Deployment Information

Data Collection

Category	Data Field	Description
Image Information	Image ID	Sha 256 for each Unique Image
Image Information	Public Download Counts	The Total Downloads for each Image
Image Information	Update Time	Exact Date time for the Last Update for each Image
Image Information	Commands/Dockerfile	The commands it runs to build the image

Data Collection (cont)

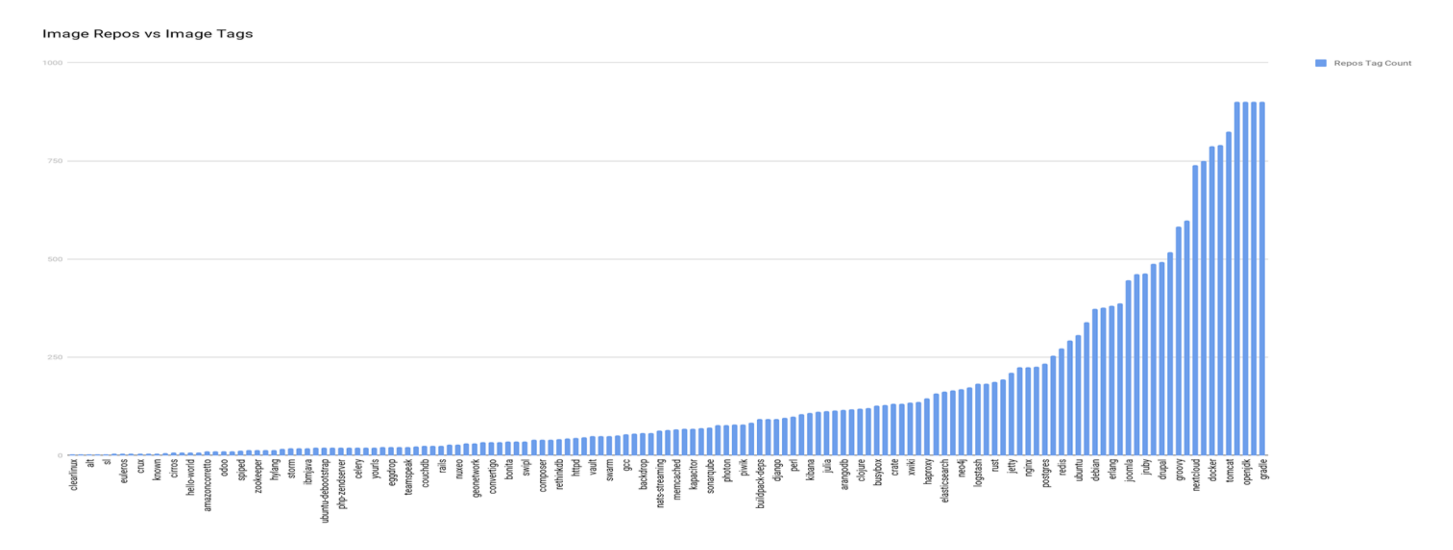
Category	Data Field	Description
Vulnerability Information	Time	First reported time and last update time
Vulnerability Information	Exploitability	How the vulnerability can be exploited (i.e, net exploitable, etc)
Vulnerability Information	Severity Ranking	CVSS 3.0/2.0, PANW Score, Signature triggers in history
Vulnerability Information	Associated Package	Name and version of vulnerable images
Deployment Information	Uses	Customized fields for images in your K8s/Openshift deployment

Basic Statistics for Dataset

Item	Statistics	Item	Statistics
# of Image Repos	151	# of Image Tags	22106
# of Dockerfile	976	# of Unique Vulnerabilities	4259
% of Vulnerable Image Repos (latest)	81%	% of Vulnerable Image Tags	69.1%

Basic Statistics for Dataset (cont)

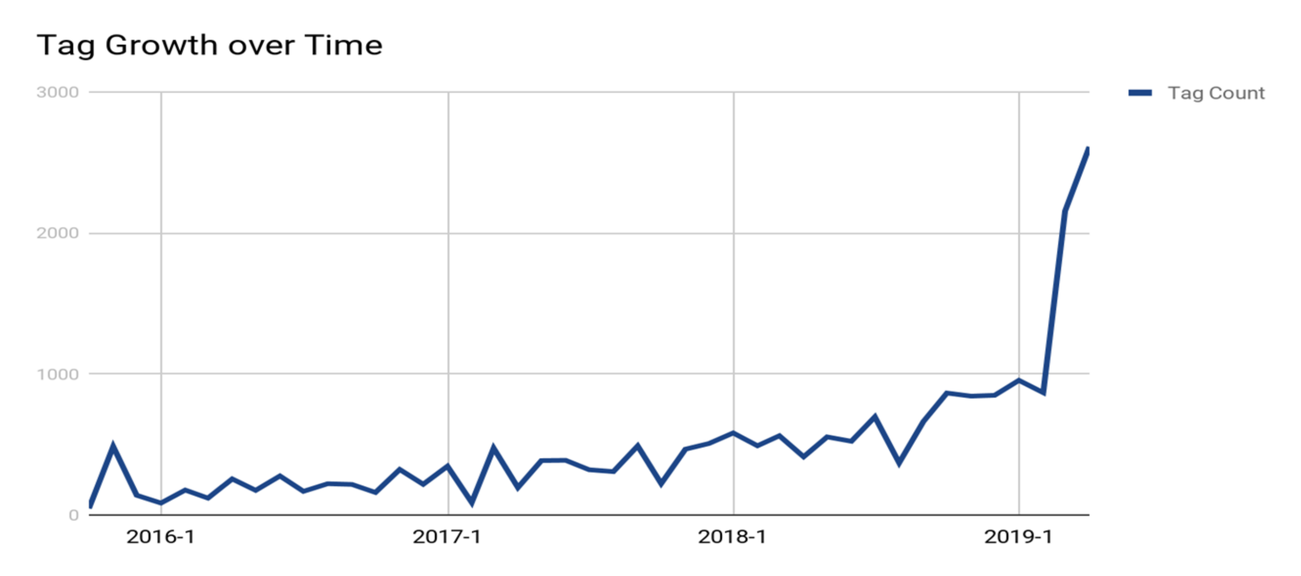
Image Repos vs Image Tags



Founding: *Popular Images has large number of Tags(Releases) and Most Repos have monthly release schedule.*

Basic Statistics for Dataset (cont)

Tag Growth over Time

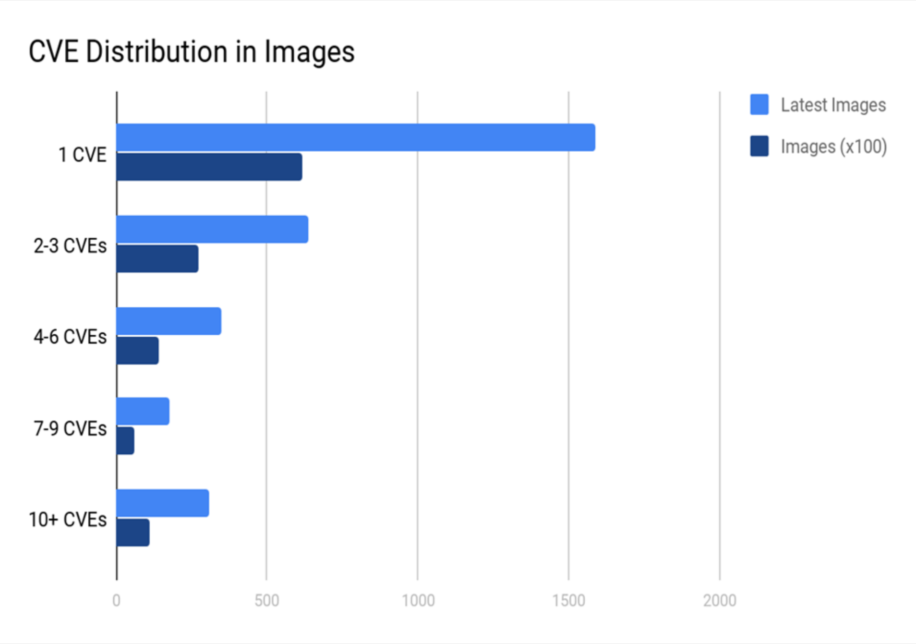


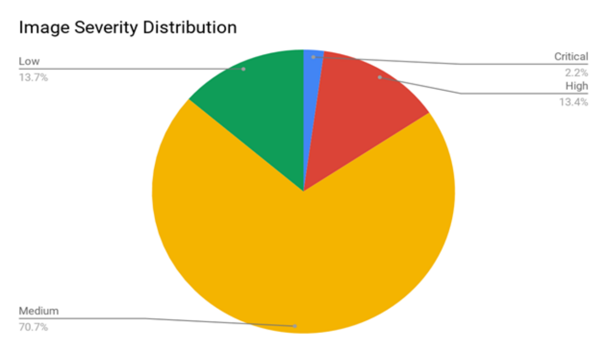
Founding: Popular Images has large number of Tags(Releases) and Most Repos have monthly release schedule.

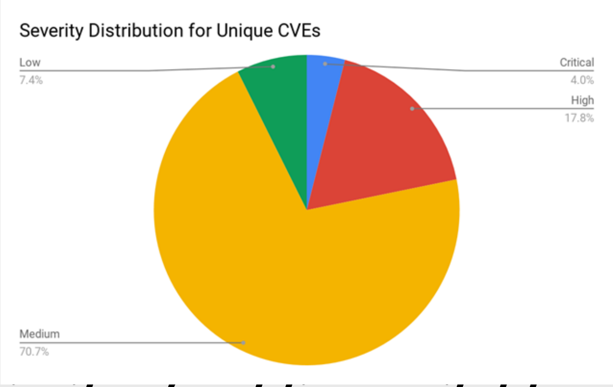
Agenda

- Introduction
- Basic Statistics
- Study of Vulnerability
- Image in your Cluster
- Practices Suggestions

Severity

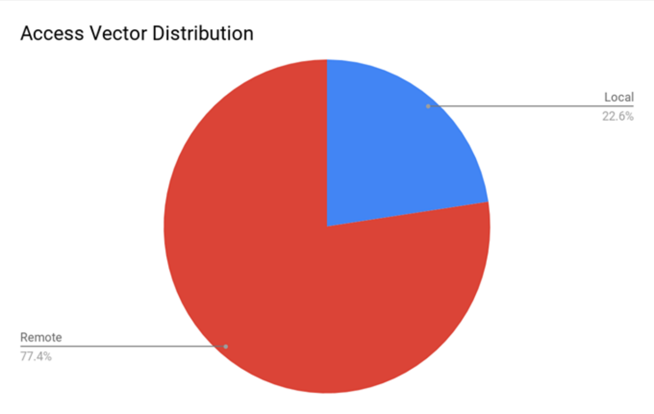


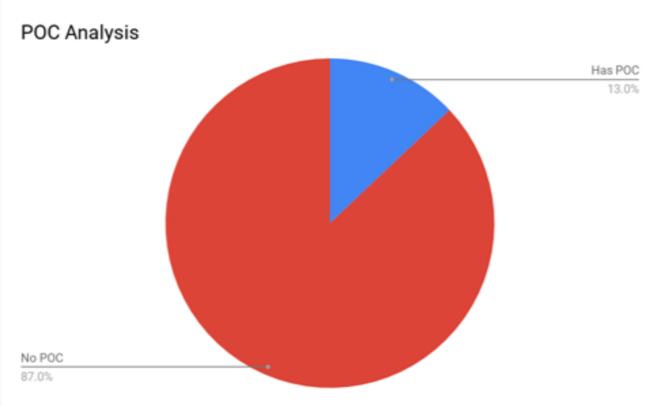


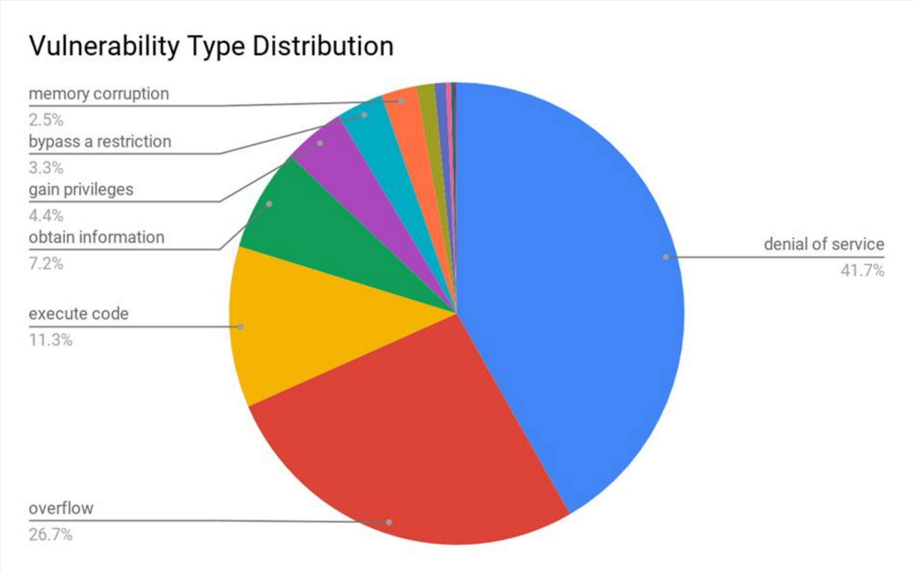


Founding: Vulnerabilities commonly exists in majority of public-available images. 81% of images have at least one vulnerability

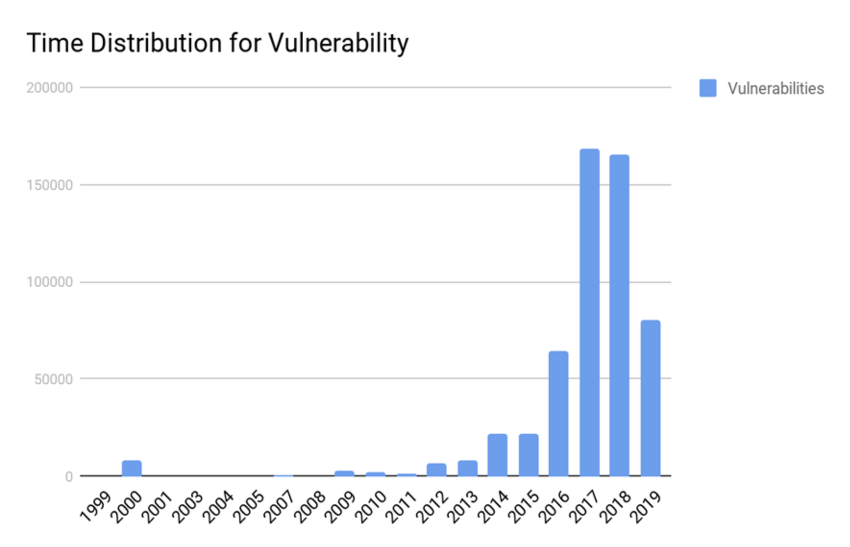
Exploitability

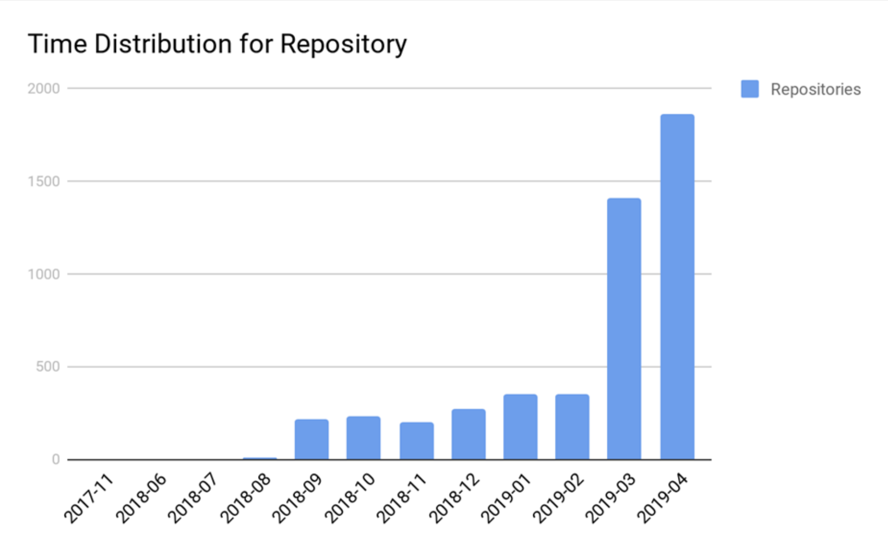




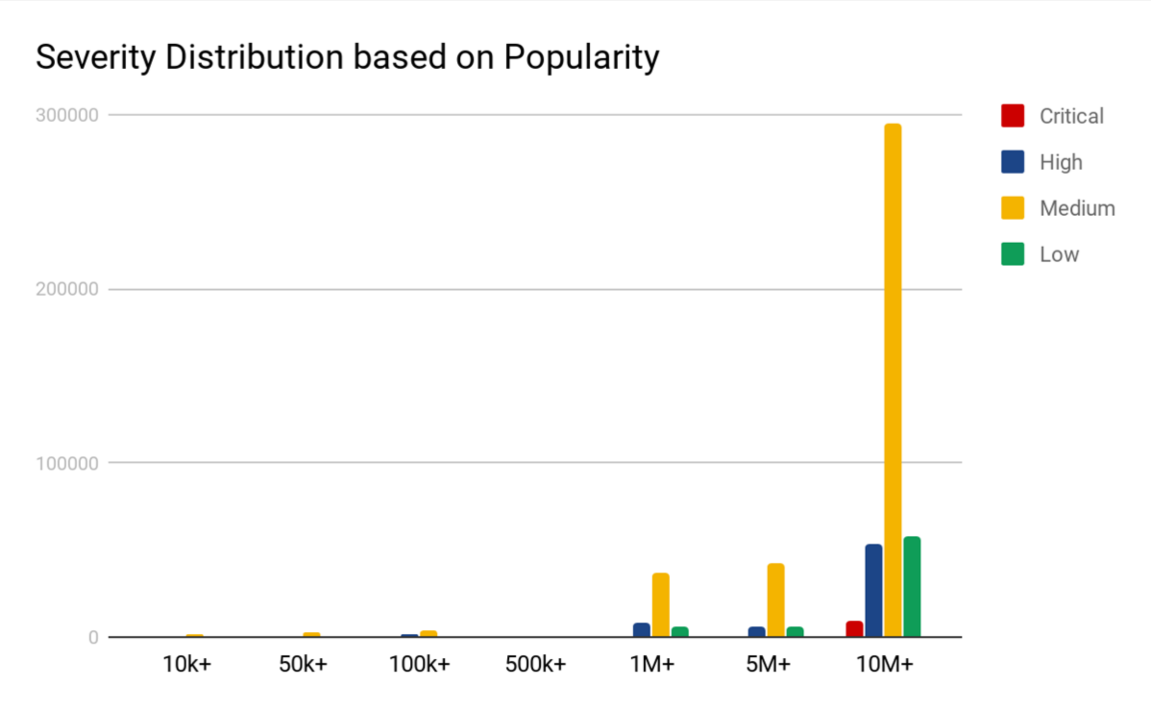


Time Sensitivity





Popularity vs Vulnerability Trend



Agenda

- ➢ Introduction
- ➢ Basic Statistics
- ➢ Study of Vulnerability
- ➢ <u>Image in your Cluster</u>
- ➢ Practices Suggestions

Risk Evaluation for K8s

Some factors we should consider for when we use vulnerable image(s) for K8s pod

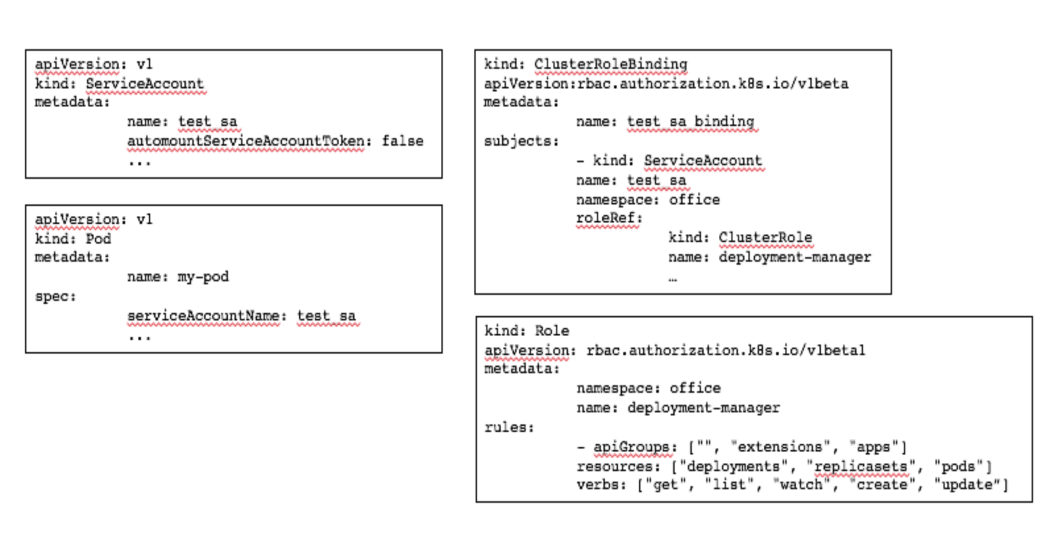
Factor	Questions	Dangerous Examples
Deployment Environment	Is the vulnerable image deployed on production namespace or not?	Vulnerable images deployed on production namespace not test namespace.
Pod Privilege	What kind of privilege we give to the vulnerable image related pod?	The pod is a privileged pod or it has been given extra capabilities.
Service Accounts Associated	What service accounts we associated with vulnerable pod/image?	The service accounts associated has ef

Risk Evaluation for K8s (cont)

Some factors we should consider for when we use vulnerable image(s) for K8s pod

Factor	Questions	Dangerous Examples
Service Exposed	Does the vulnerable pod expose external-accessible service?	The pod is exposed to internet.
Network Connected	What virtual/physical network the pod associated?	The pod can visit internet. The pod could connect to high privileged pod.

Examples



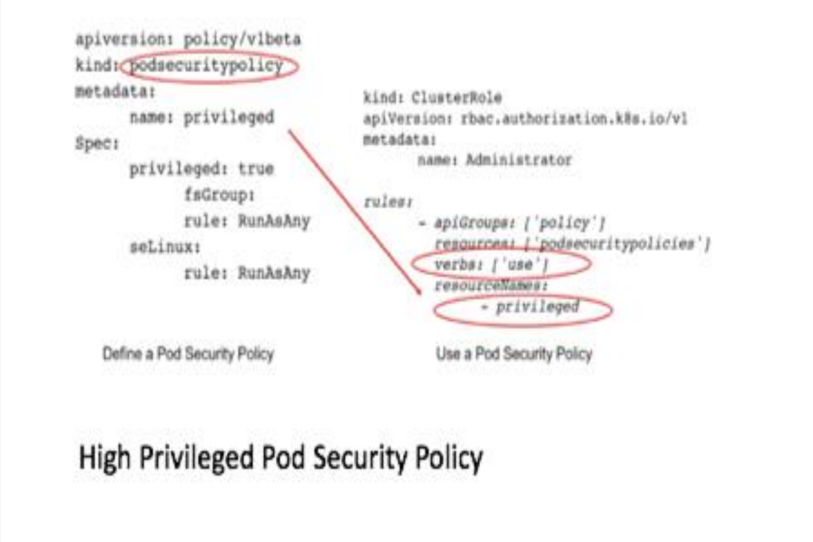
```
apiVersion: v1
kind: ServiceAccount
metadata:
          name: test sa
          automountServiceAccountToken: false
          ...
```

```
apiVersion: v1
kind: Pod
metadata:
          name: my-pod
spec:
          serviceAccountName: test sa
          ...
```

```
kind: ClusterRoleBinding
apiVersion:rbac.authorization.k8s.io/v1beta
metadata:
          name: test sa binding
subjects:
          - kind: ServiceAccount
          name: test sa
          namespace: office
          roleRef:
                    kind: ClusterRole
                    name: deployment-manager
                    …
```

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
          namespace: office
          name: deployment-manager
rules:
          - apiGroups: ["", "extensions", "apps"]
          resources: ["deployments", "replicasets", "pods"]
          verbs: ["get", "list", "watch", "create", "update"]
```

Pod with a Service Account of High Privilege



```
apiversion: policy/v1beta
kind: podsecuritypolicy
metadata:
      name: privileged
Spec:
      privileged: true
            fsGroup:
            rule: RunAsAny
      seLinux:
            rule: RunAsAny
```

Define a Pod Security Policy

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
      name: Administrator

rules:
      - apiGroups: ['policy']
        resources: ['podsecuritypolicies']
        verbs: ['use']
        resourceNames:
            - privileged
```

Use a Pod Security Policy

High Privileged Pod Security Policy

Mitigation Strategy

Solutions	How	Pros	Cons
Patch the Vulnerability	Install Patches for Officials	Solve the problem inherently	• Patch not always available in time. • Hard to patch in running container
Replace the base Image	Change the base image from Dockerfile	Easy to apply	• Secured base image not always available • Need extra test to ensure stability • Hard to change in running container
Deploy Application Firewall	Deploy an Application-level Firewall	• Prevent exploitation in runtime • Easy to apply	May not cover all vulnerabilities

Action Priority

Agenda

- ➢ Introduction
- ➢ Basic Statistics
- ➢ Study of Vulnerability
- ➢ Image in your Cluster
- ➢ <u>Practices Suggestions</u>

Container Vulnerability Management Checklist

Suggested Stage	Check Item	Action
Integration Stage	Vulnerabilities in Image	Use Image Scanner to Scan Vulnerabilities
Integration Stage	Replaceable Base Images	If base image has vulnerability, find replaceable and safe base image.
Integration Stage	Define Risk Criteria	Use criteria, such as CVSS score, exploitable vector, to define risk level for vulnerabilities

Container Vulnerability Management Checklist(cont)

Suggested Stage	Check Item	Action
Delivery Stage	Risk evaluation	Define risk tolerations criteria for your namespace/service/pod/deployment
Delivery Stage	Deployment Policy	Define policies to match vulnerabilities with deployment risk requirement
Delivery Stage	Policy Enforcement	Use tools to enforce your security policies

Container Vulnerability Management Checklist (cont)

Suggested Stage	Check Item	Action
Runtime Stage	Scan Image	Find new discovered vulnerability in running containers
Runtime Stage	Deploy an application-level firewall	Deploy an application firewall with up-to-date intrusion prevention ability
Runtime Stage	Monitor Traffic	Detect any abnormal traffic between pods using service mesh policies
Runtime Stage	Monitor Host	Deploy host-based intrusion detection to prevent host-based privilege escalation

Questions?

BLUEHAT
SHANGHAI 2019

Is my Container Secure?

Study of Vulnerability in Container World

Cecilia Hu, Yue Guan, Zhaoyan Xu

Palo Alto Networks