DevSecCon

How to Secure OpenShift Environments and What Happens If You Don´t

JAN HARRIE

London | 14-15 November 2019

$ whoami – Jan Harrie

- Security Consultant @ERNW GmbH
- Former Security Analyst/Pentester/WebApp-Monkey/Social-Engineer
- M.Sc. IT-Security TU Darmstadt

- Research // Interests:
  - K8s on-prem solutions
  - Cluster extensions
  - Gardening

@NodyTweet

Agenda

1. OpenShift & Kubernetes – Introduction & Differences
2. Cluster Threats
3. (In-)Security of Clusters
4. Conclusion & Future Work

DevSecCon

OpenShift & Kubernetes – Introduction & Differences

@NodyTweet

London | 14-15 November 2019

Introduction OpenShift

- (On-Premise) Container Execution Platform from RedHat
- First Release 05/2011
- Current Stable Release: 4.2 (11/2019)
- Host Operation System is RedHat Enterprise Linux and Container Linux from CoreOS
- Since Version 3 with K8s under the hood
- Since Version 4 Based on CRI-O, previously Docker
- OKD Community Version, e.g., CentOS
    - Current Stable Release (10/2018): v3.11
    - Builds on K8s 1.11

OpenShift vs. K8s – Differences

Kubernetes

- Role Based Access Control
- Namespaces
- Resource Limits
- Security Context
- Network Policies
- Pod Security Policies

OpenShift

- Image Streams
- Application Catalogue
- User Management
- Templates
- Revision History
- Security Context Constraints

DevSecCon

Cluster Threats

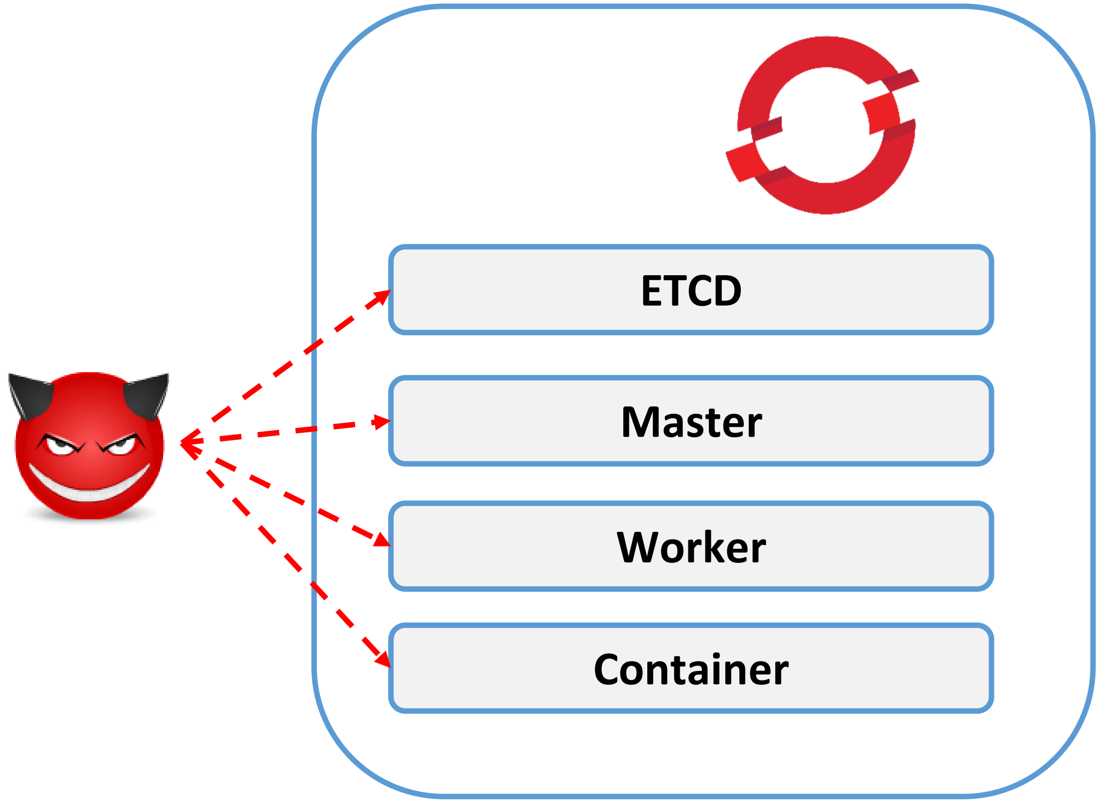
Cluster Threats

What kind of threat model exist for a cluster?

Cluster Threats

External Attacker

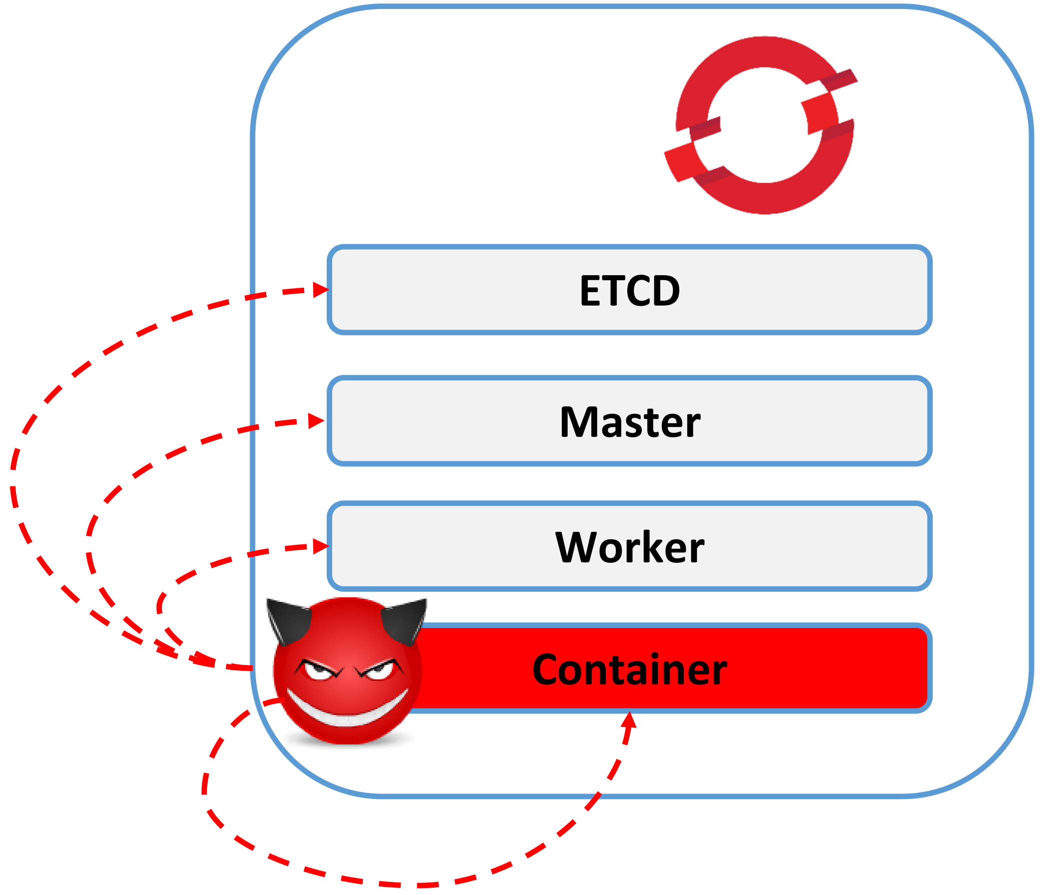
- (Only) Access to Offered Services
- No API Access
- No Cluster-Insights Knowledge
- Maybe public knowledge from DockerHub and Quay or GitHub



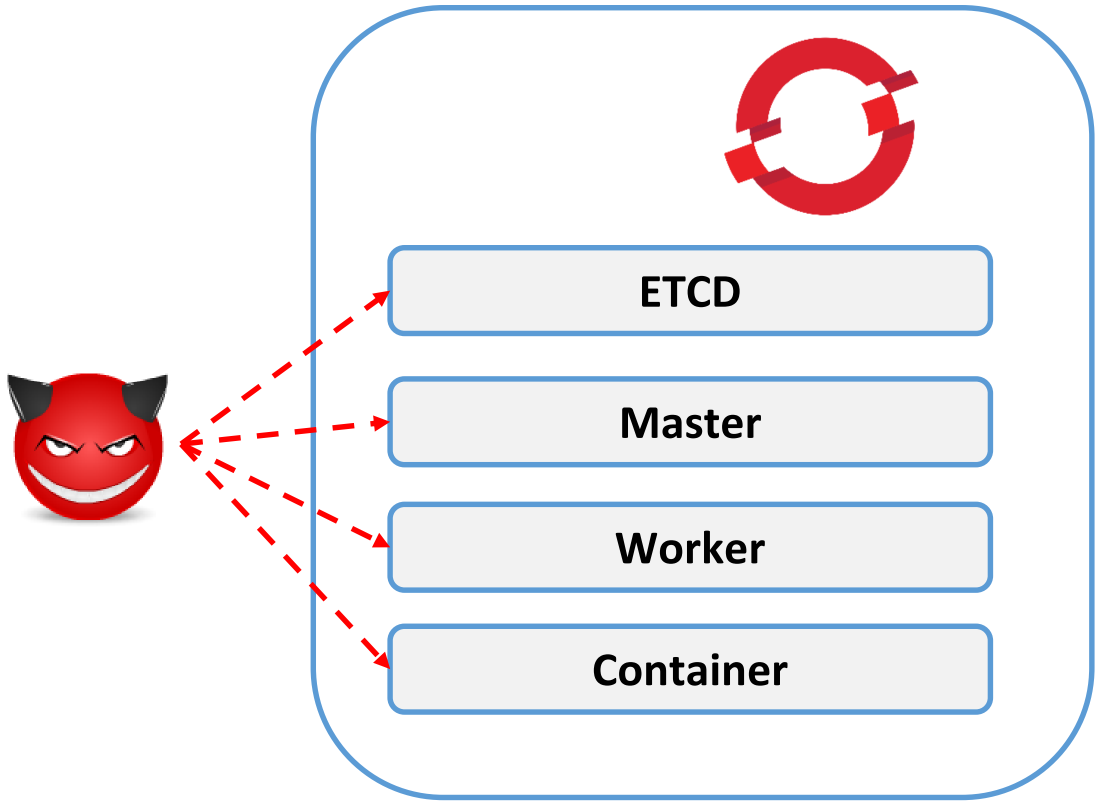
Cluster Threats

Internal Attacker

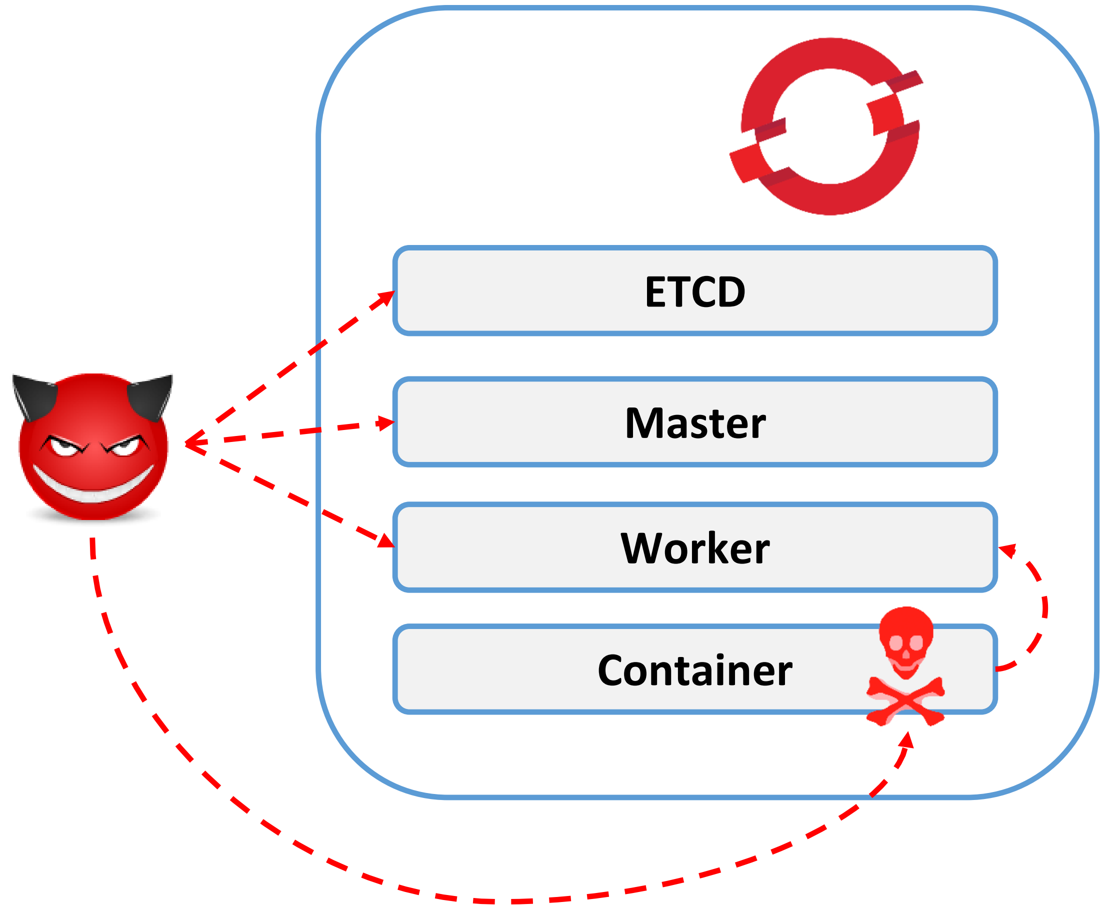
- API Access
- Control over Images and Deployments
- Access to Code Repositories
- Internal Cluster Knowledge



Internal are External Attackers one Step ahead



Internal are External Attackers one Step ahead



Internal Attacker

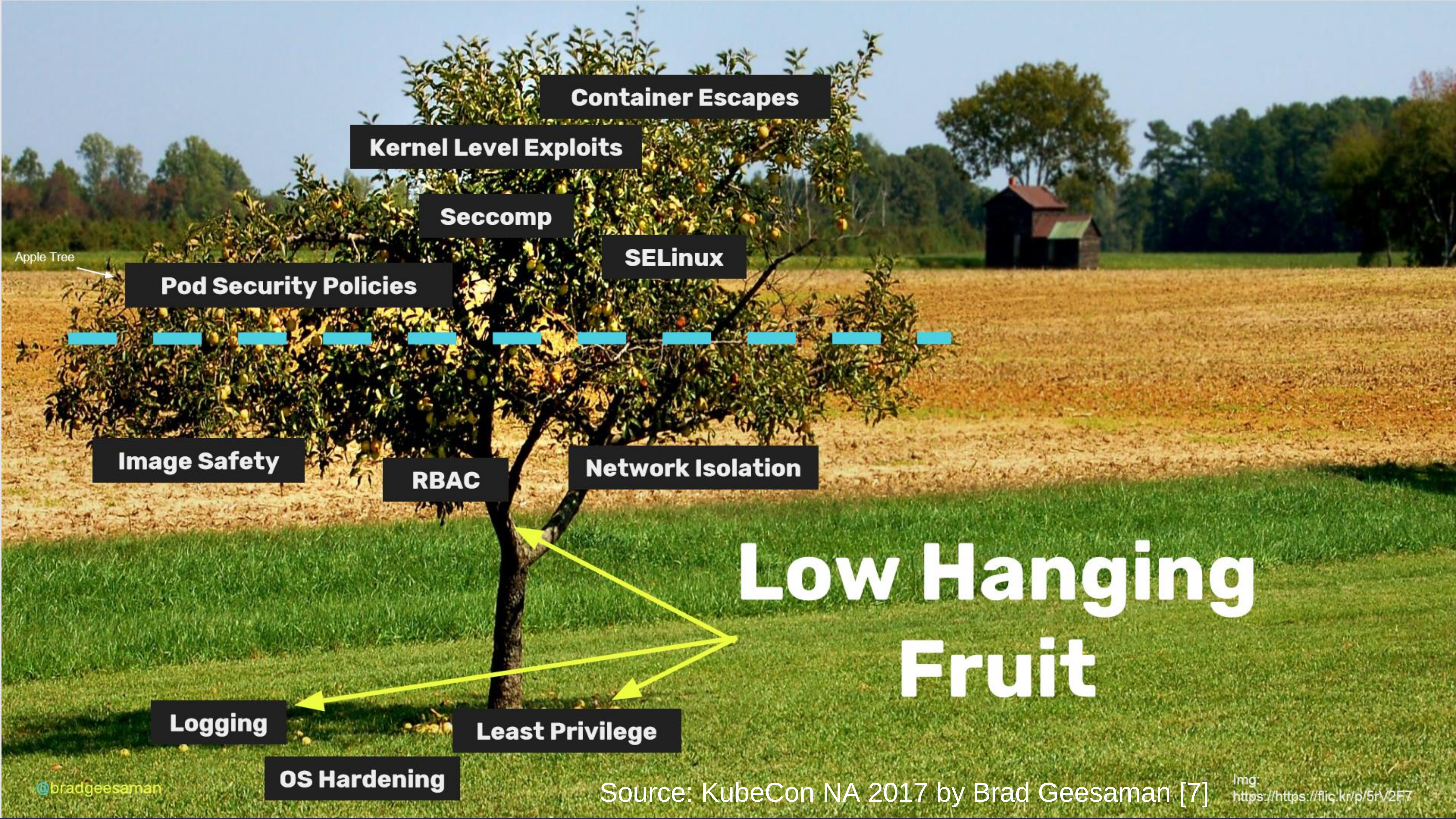
- API Access
- Control over Images and Deployments
- Access to Code Repositories
- “Cluster Internal Knowledge”

DevSecCon

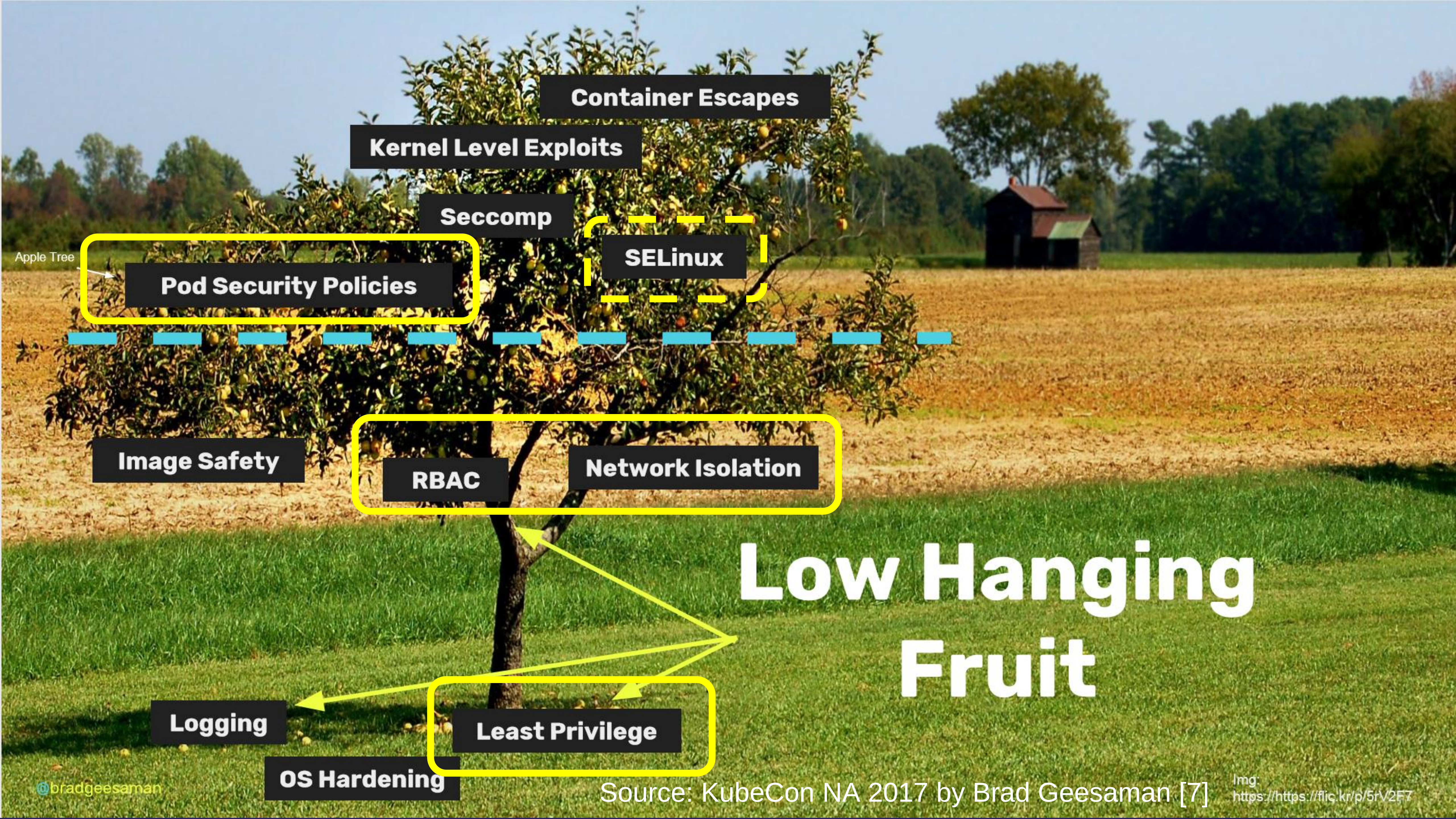
(In-)Security of Clusters

@NodyTweet

London | 14-15 November 2019



Source: KubeCon NA 2017 by Brad Geesaman [7]



Source: KubeCon NA 2017 by Brad Geesaman [7]

(In-)Security of the Cluster

User Management

Network Security

A dive into Security Context Constraints (SCC’s)

(In-)Security of the Cluster

User Management

Network Security

A dive into Security Context Constraints (SCC’s)

User Management in OpenShift

OpenShift offers integration into multiple Identity Provider (IdP)

- E.g., HTPasswd, Keystone, LDAP authentication, Basic authentication (remote), Request header, GitHub, GitLab, Google, OpenID connect ; one IdP configureable
- Implicit: `mappingMethod: claim`, Explicit: `mappingMethod: lookup`

Identities are Mapped to User in the Cluster

- Identities are bases on the IdP, while a User is an Objects in the Cluster

Users can be organized in Groups

- LDAP sync and manual assignment possible

"True User Removal" only possible in the IdP

- Manual deleted Users and Identities are re-created on next login.

Role Based Access Control

A lot of default cluster-roles are shipped with OpenShift

- → Introduction of new roles is recommended rather then adjustment
- → Modification may lead to broken functionality

Authenticated User:

- → Implicit association with virtual group `system:authenticated` // `system:authenticated:oauth`
- → What does this mean?

Demo 1: https://asciinema.org/a/281016

What can probably go wrong?

If IdP Is wrong configured:

→ Users can deploy workload in the cluster

and

→ Inspection of `resolv.conf` of the Pods:

```
$ cat /etc/resolv.conf
nameserver 172.30.0.2
search user1-p0.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

Demo 2: https://asciinema.org/a/281704

Mitigation Strategy

- Patch the Cluster Role:

```
$ oc adm policy remove-cluster-role-from-group self-provisioner system:authenticated
clusterrolebinding.rbac.authorization.k8s.io/self-provisioners patched

$ oc login -u user1
$ oc new-project user1-p1
Error from server (Forbidden): You may not request a new project via this API.
```

- Define DNS policy per Pod [12]

(In-)Security of Clusters

User Management

Network Security

A dive into Security Context Constraints (SCC’s)

Network Security

Software Defined Networking build on Open vSwitch

Three plugins available:

- `Open vSwitch Subnet`
- `Open vSwitch Multitenant`
- `Open vSwitch Networkpolicy`

- Master-Nodes do not participate in the Cluster Network
- Each Node gets its own Class-C network for the Pods assigned
- Overlay communication via VXLAN
- Integration of other Hosts into the cluster network by:
  - Host as an OpenShift node
  - Creating a VXLAN tunnel

Network Security – Open vSwitch Subnet

Configuration of **`Open vSwitch Subnet`** is not recommended

→ Cross project communication is possible

Demo 3: https://asciinema.org/a/280323

Network Security – Open vSwitch Multitenant

Setup Plugin **`Open vSwitch Multitenannt`** to “prevent” cross-project communication

- Each Project get ist own Virtual Network ID (VNID)
- Communication between different projects prohibit.
- Projects can be joined together

BUT !

- Separation on Namespace-Level
- Projects with ***VNID 0*** are more privileged
- The project ***default*** has ***VNID 0***

Side reference: TR19 – VXLAN Security or Injection [8]

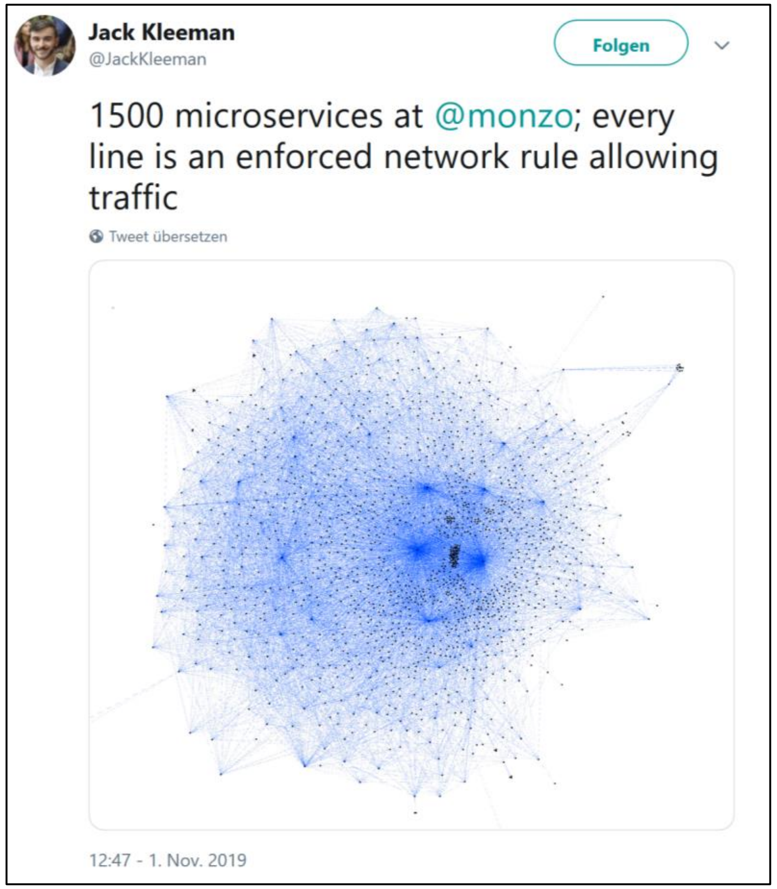
Network Security – Open vSwitch Networkpolicy

Alternatively: stick to ovs-networkpolicy which allows you to deploy NetworkPolicies, and bock all ingress traffic [9] and add explicit whitelistings.

```
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

Further more, the plugin allows White- an Black-Listing on Layer3 [10] with CIDR notation or DNS

Configuration of Egress IP’s and Egress Proxies is possible [11]


Jack Kleeman
@JackKleeman
Folgen

1500 microservices at @monzo; every line is an enforced network rule allowing traffic

Tweet übersetzen

12:47 - 1. Nov. 2019

https://twitter.com/JackKleeman/status/1190354757308862468

(In-)Security of Clusters

User Management

Network Security

A dive into Security Context Constraints (SCC’s)

Security Context Constraints

- Introduced by release 3.0 (05/2015)
- Secure Context Constraints (SCC’s) is for Pods what RBAC is for the SAs
- Restrict execution of Pods
- Created by Cluster Administrator and assigned to Service Account
- Default SCC is ‘restricted’

Security Context Constraints

Predefines Profiles

```
$ oc get scc
```

```
NAME              PRIV     CAPS     SELINUX     RUNASUSER          [...]
anyuid            false    []       MustRunAs   RunAsAny           [...]
hostaccess        false    []       MustRunAs   MustRunAsRange     [...]
hostmount-anyuid  false    []       MustRunAs   RunAsAny           [...]
hostnetwork       false    []       MustRunAs   MustRunAsRange     [...]
nonroot           false    []       MustRunAs   MustRunAsNonRoot   [...]
privileged        true     [*]      RunAsAny    RunAsAny           [...]
restricted        false    []       MustRunAs   MustRunAsRange     [...]
```

Security Context Constraints

Predefines Profiles – that allow **privileged**

```
$ oc get scc

NAME                PRIV     CAPS     SELINUX     RUNASUSER          [...]
anyuid              false    []       MustRunAs   RunAsAny           [...]
hostaccess          false    []       MustRunAs   MustRunAsRange     [...]
hostmount-anyuid    false    []       MustRunAs   RunAsAny           [...]
hostnetwork         false    []       MustRunAs   MustRunAsRange     [...]
nonroot             false    []       MustRunAs   MustRunAsNonRoot   [...]
privileged          true     [*]      RunAsAny    RunAsAny           [...]
restricted          false    []       MustRunAs   MustRunAsRange     [...]
```

Demo 4: https://asciinema.org/a/280383

Security Context Constraints

Predefines Profiles – that allow **hostPath, hostIPC, hostPID**

```
$ oc get scc

NAME                PRIV     CAPS     SELINUX     RUNASUSER          [...]
anyuid              false    []       MustRunAs   RunAsAny           [...]
hostaccess          false    []       MustRunAs   MustRunAsRange     [...]
hostmount-anyuid    false    []       MustRunAs   RunAsAny           [...]
hostnetwork         false    []       MustRunAs   MustRunAsRange     [...]
nonroot             false    []       MustRunAs   MustRunAsNonRoot   [...]
privileged          true     [*]      RunAsAny    RunAsAny           [...]
restricted          false    []       MustRunAs   MustRunAsRange     [...]
```

Demo 5: https://asciinema.org/a/280388

Security Context Constraints

Predefines Profiles – that allow **root in container**

```
$ oc get scc
```

```
NAME                 PRIV      CAPS      SELINUX     RUNASUSER          [...]
anyuid               false     []        MustRunAs   RunAsAny           [...]
hostaccess           false     []        MustRunAs   MustRunAsRange     [...]
hostmount-anyuid     false     []        MustRunAs   RunAsAny           [...]
hostnetwork          false     []        MustRunAs   MustRunAsRange     [...]
nonroot              false     []        MustRunAs   MustRunAsNonRoot   [...]
privileged           true      [*]       RunAsAny    RunAsAny           [...]
restricted           false     []        MustRunAs   MustRunAsRange     [...]
```

Security Context Constraints

Predefines Profiles – available

```
$ oc get scc

NAME               PRIV    CAPS    SELINUX     RUNASUSER          [...]
anyuid             false   []      MustRunAs   RunAsAny           [...]
hostaccess         false   []      MustRunAs   MustRunAsRange     [...]
hostmount-anyuid   false   []      MustRunAs   RunAsAny           [...]
hostnetwork        false   []      MustRunAs   MustRunAsRange     [...]
nonroot            false   []      MustRunAs   MustRunAsNonRoot   [...]
privileged         true    [*]     RunAsAny    RunAsAny           [...]
restricted         false   []      MustRunAs   MustRunAsRange     [...]
```

Security Context Constraints

Predefines Profiles – available

```
$ oc get scc

NAME                PRIV     CAPS     SELINUX     RUNASUSER          [...]
anyuid              false    []       MustRunAs   RunAsAny           [...]
hostaccess          false    []       MustRunAs   MustRunAsRange     [...]
hostmount-anyuid    false    []       MustRunAs   RunAsAny           [...]
hostnetwork         false    []       MustRunAs   MustRunAsRange     [...]
nonroot             false    []       MustRunAs   MustRunAsNonRoot   [...]
privileged          true     [*]      RunAsAny    RunAsAny           [...]
restricted          false    []       MustRunAs   MustRunAsRange     [...]
```

Security Context Constraints – Summary

- Integration of SELinux is great benefit
- Do not use existing Security Context Constraints except:
  - `restricted`
  - `nonroot`
- Create dedicated SCC's with least privilege principle if necessary

Demo 6: https://asciinema.org/a/280392

Source: [13]

DevSecCon

Conclusion & Future Work

Conclusion & Future Work

- OpenShift raises the bar by it’s defaults, but must be further adjusted
- Quick releases with feature extension/adjustment challenges the security research
- Multiple components are dedicated developed by RedHat and are not spread for the community
- OpenShift 4.2 is already available and components have been refactored and, new features and concepts are available

DevSecCon

Thanks for you Attention and take care of your defaults ;)

References

[1] https://blog.openshift.com/introducing-red-hat-openshift-4-2-developers-get-an-expanded-and-improved-toolbox/

[2] https://twitter.com/bradgeesaman/status/1188541320626921473

[3] https://blog.openshift.com/red-hat-chose-kubernetes-openshift/

[4]https://kubernetes.io/blog/2015/04/borg-predecessor-to-kubernetes/

[5] https://ai.google/research/pubs/pub43438

[6] https://twitter.com/iancoldwater/status/1174793280011325456

[7] https://goo.gl/TNRxtd

[8] https://troopers.de/downloads/troopers19/TROOPERS19_AR_VXLAN_Security.pdf

[9] https://kubernetes.io/docs/concepts/services-networking/network-policies/#default-deny-all-ingress-traffic

[10] https://docs.openshift.com/container-platform/3.11/admin_guide/managing_networking.html#admin-guide-limit-pod-access-egress

[11] https://docs.openshift.com/container-platform/3.11/admin_guide/managing_networking.html#admin-guide-limit-pod-access-egress-router

[12] https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/#pod-s-dns-policy

[13] https://cookbook.openshift.org/users-and-role-based-access-control/how-can-i-enable-an-image-to-run-as-a-set-user-id.html