

7 июня 2023 📍 Москва, МЦК ЗИЛ

БЕКОН²³

Первая в России конференция
по БЕзопасности КОНтейнеров и контейнерных сред

Не такой очевидный RBAC Kubernetes

Дмитрий Евдокимов

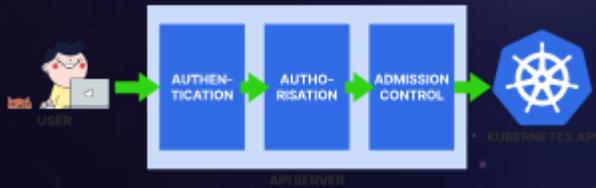
Founder&CTO Luntry

- Основатель и технический директор [Luntry](#)
- Опыт в ИБ более 10 лет
- CFP ZeroNights, DevOpsConf
- Бывший автор статей и редактор рубрик в журнале “ХАКЕР”
- Автор Telegram-канала [📩 “k8s \(in\)security”](#)
- Автор курса “Cloud Native безопасность в Kubernetes”
- Не верит, что систему можно сделать надежной и безопасной, не понимая ее
- Докладчик: BlackHat, HITB, ZeroNights, HackInParis, Confidence, SAS, OFFZONE, PHDays, Kazhackstan, DevOpsConf, KuberConf, VK Kubernetes Conference, HighLoad++ и др.



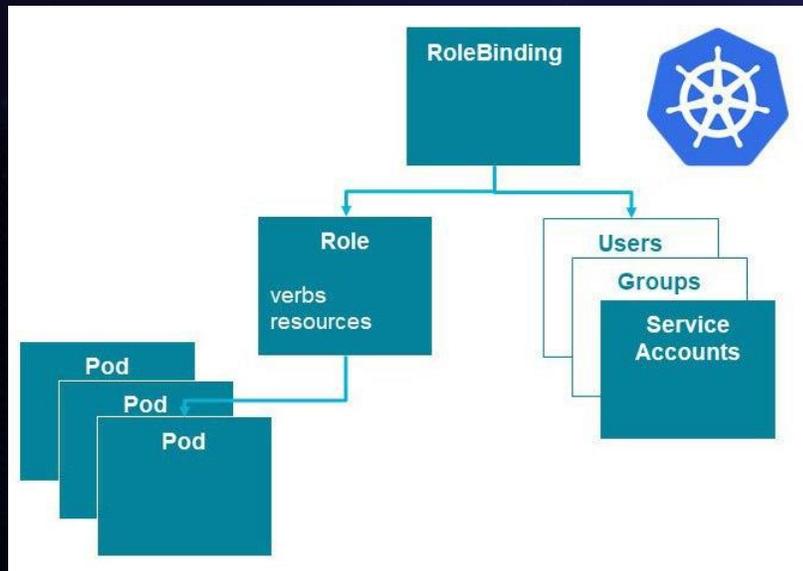
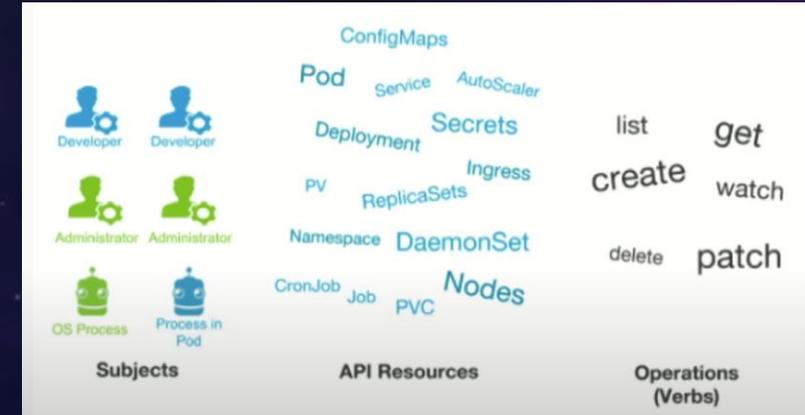


Введение



Authorization: Mechanisms

Mechanism	Decision source	Usage
Node	API Server built-in	Internal use (kubelets)
ABAC	Static file	Insecure, deprecated
RBAC	API Objects	Users and administrators
WebHook	External services	Integration
AlwaysDeny	API Server built-in	Testing
AlwaysAllow		



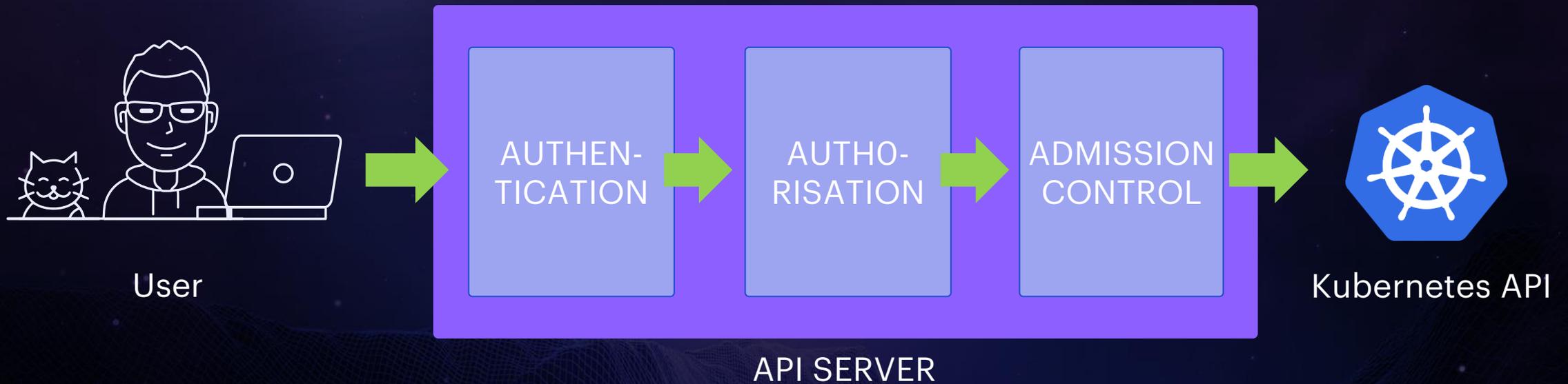
```

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
    
```

```

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
    
```

RBAC с высоты птичьего полеты

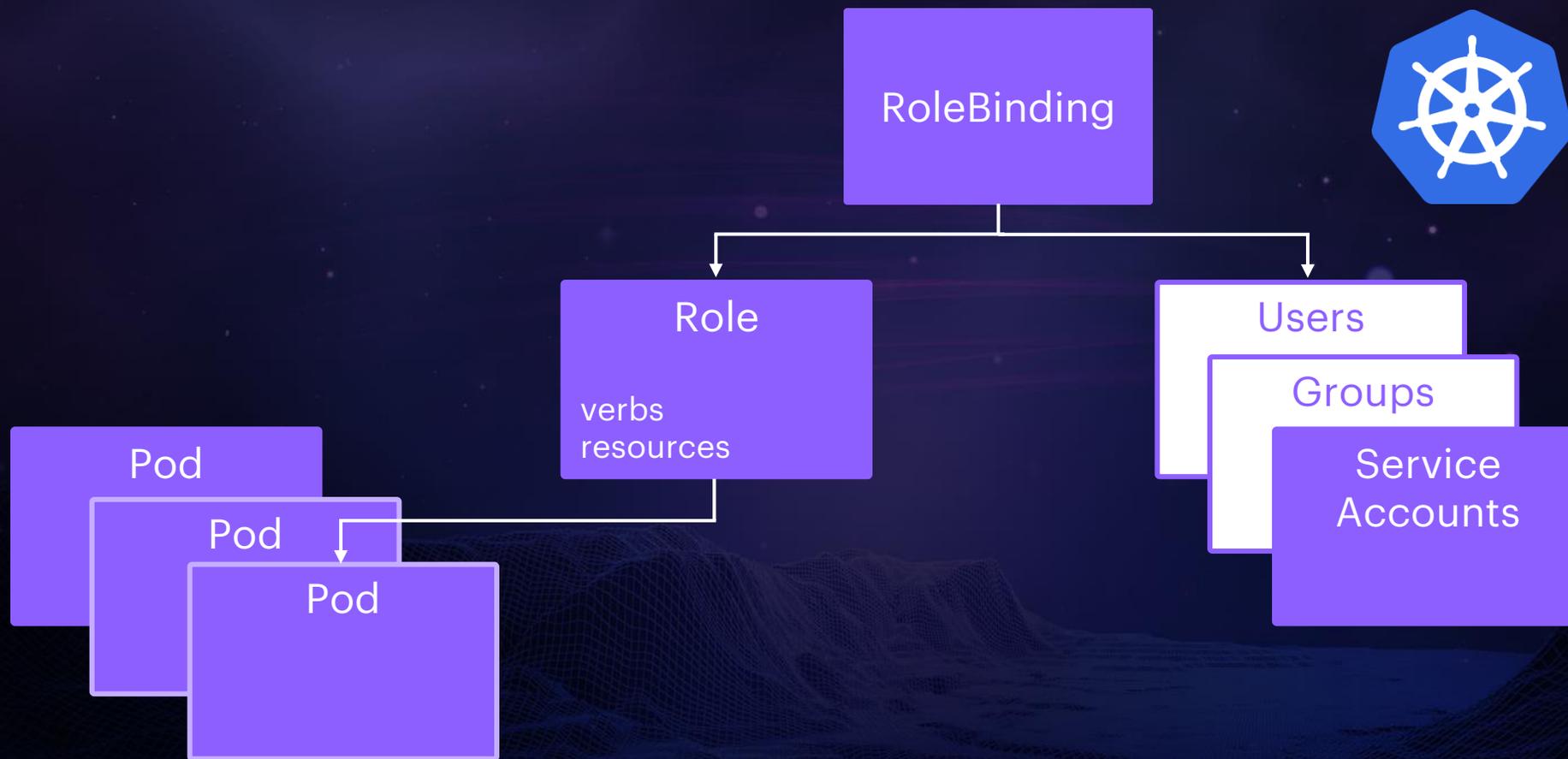


Authorization: Mechanisms

Mechanism	Decision source
Node	API Server built-in
ABAC	Static file
RBAC	API Objects
WebHook	External services
AlwaysDeny AlwaysAllow	API Server built-in

RWAS с высоты птичьего полеты

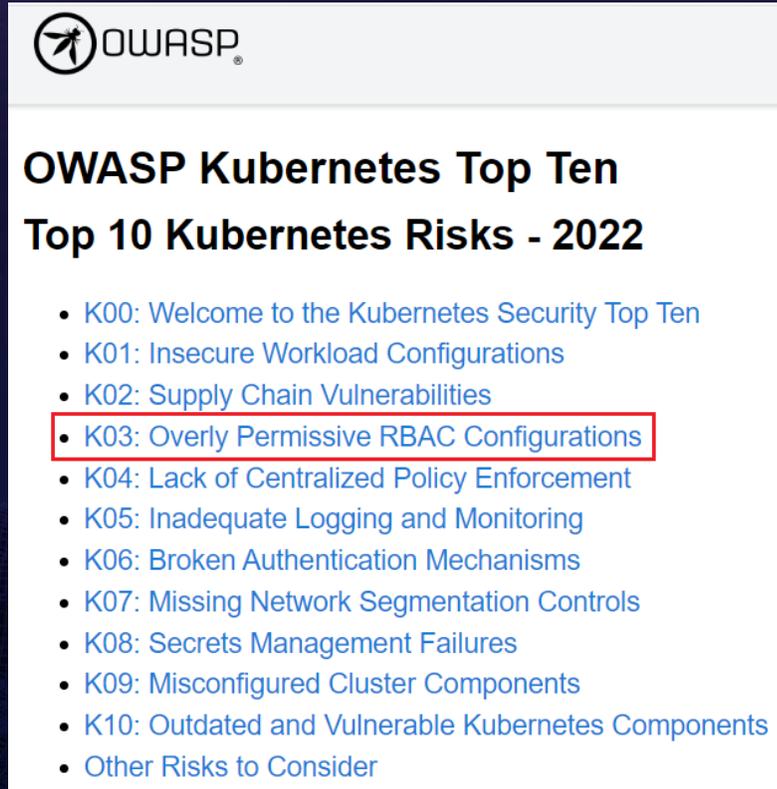




Проводите аудит RBAC ?

Опасные права можно использовать для следующих атак:

- Manipulate AuthN/Z
- Acquire Tokens
- RCE
- Steal Pods
- Meddler-in-the-Middle

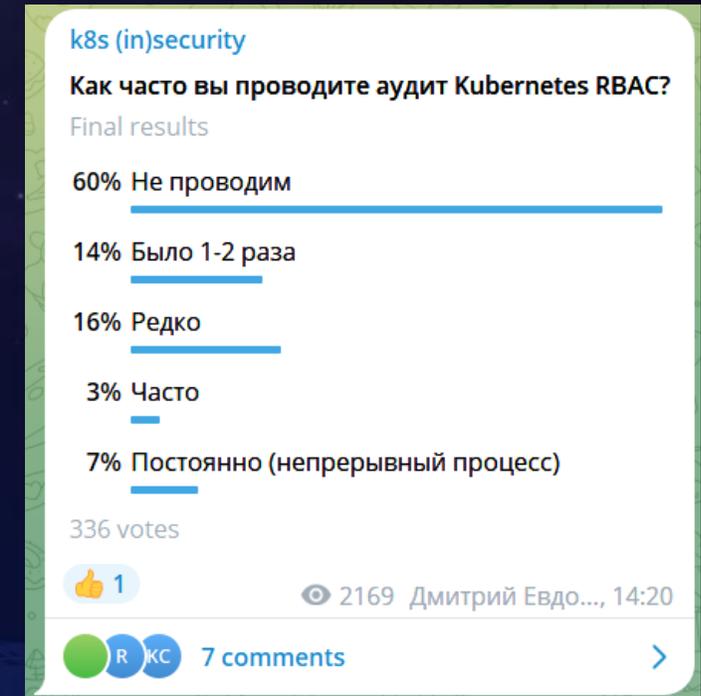


OWASP

OWASP Kubernetes Top Ten

Top 10 Kubernetes Risks - 2022

- K00: Welcome to the Kubernetes Security Top Ten
- K01: Insecure Workload Configurations
- K02: Supply Chain Vulnerabilities
- **K03: Overly Permissive RBAC Configurations**
- K04: Lack of Centralized Policy Enforcement
- K05: Inadequate Logging and Monitoring
- K06: Broken Authentication Mechanisms
- K07: Missing Network Segmentation Controls
- K08: Secrets Management Failures
- K09: Misconfigured Cluster Components
- K10: Outdated and Vulnerable Kubernetes Components
- Other Risks to Consider



Источник

Пример: RBAC от ArgoCD и Flux

В системе могут появляться опасные Role/ClusterRole

В системе могут появляться RoleBinding/ClusterRoleBinding нас стандартные Role/ClusterRole с опасными правами

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  labels:
    name: flux
  name: flux
rules:
  - apiGroups: ['*']
    resources: ['*']
    verbs: ['*']
  - nonResourceURLs: ['*']
    verbs: ['*']
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: argocd-application-controller
    app.kubernetes.io/part-of: argocd
    app.kubernetes.io/component: application-controller
  name: argocd-application-controller
rules:
  - apiGroups:
    - '*'
    resources:
    - '*'
    verbs:
    - '*'
  - nonResourceURLs:
    - '*'
    verbs:
    - '*'
```

Тонкие моменты

Момент 1: встроенные сущности

- Встроенные сущности описаны прямо в [коде](#): /plugin/pkg/auth/authorizer/rbac/bootstrappolicy/policy.go
- `kubectl get roles,rolebindings,clusterroles,clusterrolebindings -A -l kubernetes.io/bootstrapping=rbac-defaults`

```
// ClusterRoles returns the cluster roles to bootstrap an API server with
func ClusterRoles() []rbacv1.ClusterRole {
    roles := []rbacv1.ClusterRole{
        {
            // a "root" role which can do absolutely anything
            ObjectMeta: metav1.ObjectMeta{Name: "cluster-admin"},
            Rules: []rbacv1.PolicyRule{
                rbacv1helpers.NewRule("**").Groups("**").Resources("**").RuleOrDie(),
                rbacv1helpers.NewRule("**").URLs("**").RuleOrDie(),
            },
        },
    },
}
```

```
~/Documents kubectl get roles,rolebindings,clusterroles,clusterrolebindings -A -l kubernetes.io/bootstrapping=rbac-defaults
NAMESPACE NAME CREATED AT
kube-public role.rbac.authorization.k8s.io/system:controller:bootstrap-signer 2023-01-30T12:48:13Z
kube-system role.rbac.authorization.k8s.io/extension-apiserver-authentication-reader 2023-01-30T12:48:13Z
kube-system role.rbac.authorization.k8s.io/system:leader-locking-kube-controller-manager 2023-01-30T12:48:13Z
kube-system role.rbac.authorization.k8s.io/system:leader-locking-kube-scheduler 2023-01-30T12:48:13Z
kube-system role.rbac.authorization.k8s.io/system:controller:bootstrap-signer 2023-01-30T12:48:13Z
kube-system role.rbac.authorization.k8s.io/system:controller:cloud-provider 2023-01-30T12:48:13Z
kube-system role.rbac.authorization.k8s.io/system:controller:token-cleaner 2023-01-30T12:48:13Z

NAMESPACE AGE NAME ROLE
kube-public 4h20m rolebinding.rbac.authorization.k8s.io/system:controller:bootstrap-signer Role/system:controller:bootstrap-signer
kube-system 4h20m rolebinding.rbac.authorization.k8s.io/system:extension-apiserver-authentication-reader Role/extension-apiserver-authentication-read
er 4h20m
kube-system 4h20m rolebinding.rbac.authorization.k8s.io/system:leader-locking-kube-controller-manager Role/system:leader-locking-kube-controller-
manager 4h20m
kube-system 4h20m rolebinding.rbac.authorization.k8s.io/system:leader-locking-kube-scheduler Role/system:leader-locking-kube-scheduler
4h20m
kube-system 4h20m rolebinding.rbac.authorization.k8s.io/system:controller:bootstrap-signer Role/system:controller:bootstrap-signer
4h20m
kube-system 4h20m rolebinding.rbac.authorization.k8s.io/system:controller:cloud-provider Role/system:controller:cloud-provider
4h20m
kube-system 4h20m rolebinding.rbac.authorization.k8s.io/system:controller:token-cleaner Role/system:controller:token-cleaner
4h20m

NAMESPACE NAME CREATED AT
clusterrole.rbac.authorization.k8s.io/admin 2023-01-30T12:48:13Z
clusterrole.rbac.authorization.k8s.io/cluster-admin 2023-01-30T12:48:13Z
clusterrole.rbac.authorization.k8s.io/edit 2023-01-30T12:48:13Z
clusterrole.rbac.authorization.k8s.io/system:aggregate-to-admin 2023-01-30T12:48:13Z
```

	1.16.15	1.22.17	1.27.2
Role	7	7	7
RoleBinding	7	7	7
ClusterRole	53	62	62
ClusterRoleBinding	36	43	43

Момент 3: аддитивная природа прав

- Права в системе можно только добавлять
- Забирать/ограничивать можно через PolicyEngine
- Реальные/полные права субъекта можно увидеть только в Runtime – в процессе выкатки YAML мы видим только его один



- Нет фиксированного списка verbs
- Существуют virtual verbs
 - use
 - bind
 - sign
 - proxy
 - escalate
 - impersonate
 - userextras
- Если вы допустите опечатку, ничего не сломается!
- Можно создавать свои права доступа (verbs) ;)
- Пример: [“RBAC Virtual Verbs: Teaching Kubernetes to Educate Dolphins”](#)

HTTP verb	request verb
POST	create
GET, HEAD	get (for individual resources), list (for collections, including full object content), watch (for watching an individual resource or collection of resources)
PUT	update
PATCH	patch
DELETE	delete (for individual resources), deletecollection (for collections)

Kubernetes sometimes checks authorization for additional permissions using specialized verbs. For example:

- **PodSecurityPolicy**
 - use verb on podsecuritypolicies resources in the policy API group.
- **RBAC**
 - bind and escalate verbs on roles and clusterroles resources in the rbac.authorization.k8s.io API group.
- **Authentication**
 - impersonate verb on users, groups, and serviceaccounts in the core API group, and the userextras in the authentication.k8s.io API group.

[Источник](#)

Listing secrets

It is generally clear that allowing `get` access on Secrets will allow a user to read their contents. It is also important to note that `list` and `watch` access also effectively allow for users to reveal the Secret contents. For example, when a List response is returned (for example, via `kubectl get secrets -A -o yaml`), the response includes the contents of all Secrets.

Find LIST Secrets with no GET in k8s:

```
kubectl get clusterroles -o json | \
  jq '(rules[] |
  index("secrets"))
  | select(. == null) | not))) |'
```

For these reasons `watch` and `list` requests for secrets within a namespace are extremely powerful capabilities and should be avoided, since listing secrets allows the clients to inspect the values of all secrets that are in that namespace. The ability to `watch` and `list` all secrets in a cluster should be reserved for only the most privileged, system-level components.

Момент 5: Subresources, не существующие ресурсы

```
ubuntu@ip-172-31-40-152:~$ kubectl api-resources
NAME                SHORTNAMES  APIVERSION          NAMESPACE  KIND
bindings            v1          v1                  true        Binding
componentstatuses   cs          v1                  false       ComponentStatus
configmaps          cm          v1                  true        ConfigMap
endpoints           ep          v1                  true        Endpoints
events              ev          v1                  true        Event
limitranges         limits      v1                  true        LimitRange
namespaces          ns          v1                  false       Namespace
nodes               no          v1                  false       Node
persistentvolumeclaims  pvc        v1                  true        PersistentVolumeClaim
persistentvolumes    pv          v1                  false       PersistentVolume
pods                po          v1                  true        Pod
podtemplates        v1          v1                  true        PodTemplate
replicationcontrollers  rc          v1                  true        ReplicationController
resourcequotas      quota       v1                  true        ResourceQuota
secrets             v1          v1                  true        Secret
serviceaccounts     sa          v1                  true        ServiceAccount
services            svc          v1                  true        Service
mutatingwebhookconfigurations  admissionregistration.k8s.io/v1  false       MutatingWebhookConfiguration
validatingwebhookconfigurations  admissionregistration.k8s.io/v1  false       ValidatingWebhookConfiguration
customresourcedefinitions  crd,crds  apiextensions.k8s.io/v1  false       CustomResourceDefinition
apiservices          v1          apiregistration.k8s.io/v1  false       APIService
controllerrevisions  v1          apps/v1             true        ControllerRevision
daemonsets           ds          apps/v1             true        DaemonSet
deployments          deploy     apps/v1             true        Deployment
replicasets          rs          apps/v1             true        ReplicaSet
statefulsets         sts          apps/v1             true        StatefulSet
tokenreviews          v1          authentication.k8s.io/v1  false       TokenReview
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: edit
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - pods/attach
  - pods/exec
  - pods/portforward
  - pods/proxy
  verbs:
  - create
  - delete
  - deletecollection
  - patch
  - update
```

Получаем все subresources

- Простенький [скрипт](#)
- В 1.27.2 есть 41 subresources

```
#!/bin/bash

list=$(kubectl get --raw / |grep "^ \"/api"|sed 's/[",,]//g');
for _api in ${list[@]}; do
  _aruyo=$(kubectl get --raw $_api | jq .resources);
  if [ "x${_aruyo}" != "xnull" ]; then
    echo;
    echo "===${_api}===";
    kubectl get --raw $_api | jq -r ".resources[].name";
  fi;
done
```

```
namespaces/finalize
namespaces/status
nodes/proxy
nodes/status
persistentvolumeclaims/status
persistentvolumes/status
pods/attach
pods/binding
pods/ephemeralcontainers
pods/eviction
pods/exec
pods/log
pods/portforward
pods/proxy
pods/status
replicationcontrollers/scale
replicationcontrollers/status
resourcequotas/status
serviceaccounts/token
services/proxy
services/status
customresourcedefinitions/status
```

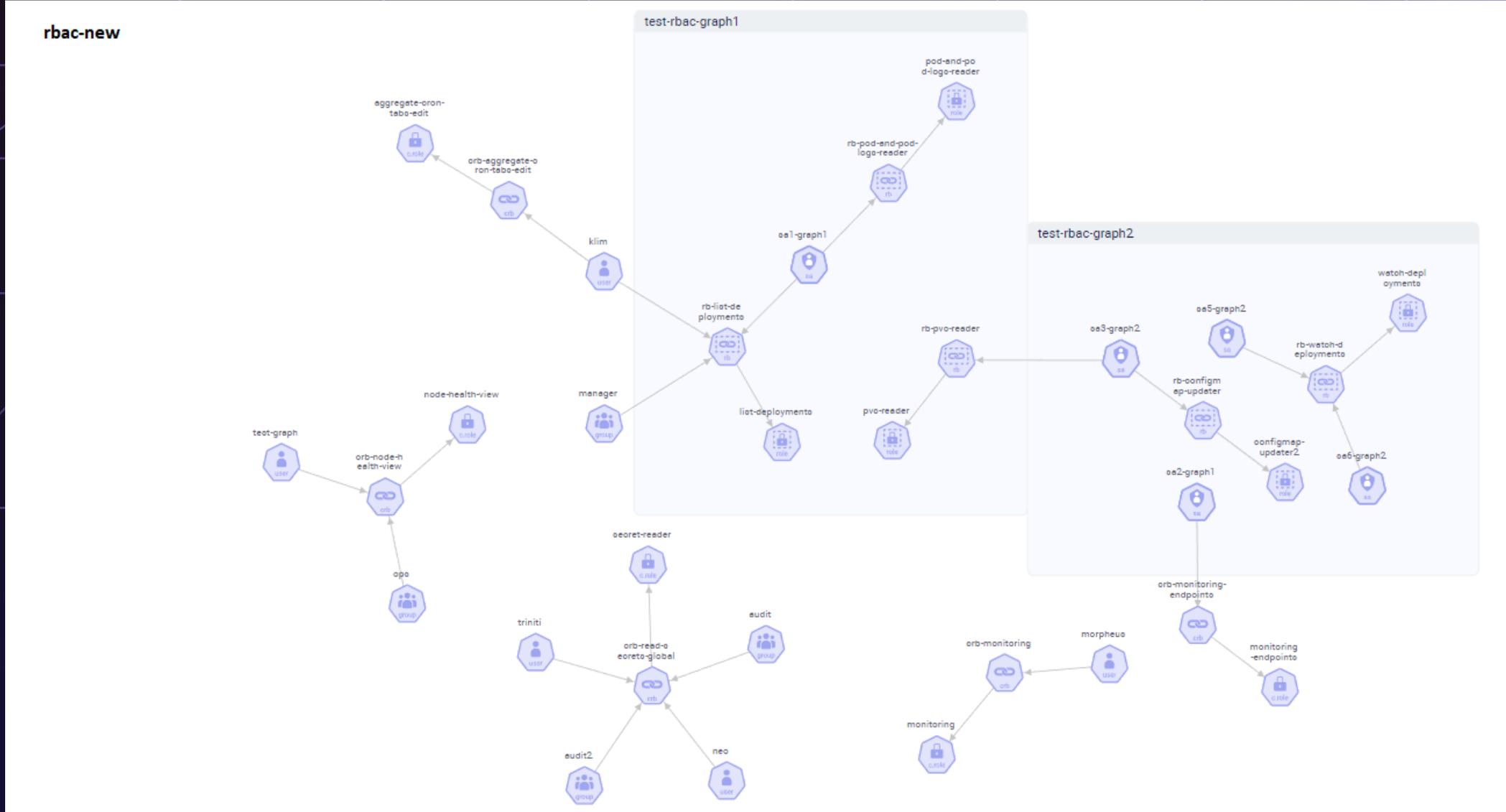
```
apiservices/status
daemonsets/status
deployments/scale
deployments/status
replicasets/scale
replicasets/status
statefulsets/scale
statefulsets/status
horizontalpodautoscalers/status
cronjobs/status
jobs/status
certificatesigningrequests/approval
certificatesigningrequests/status
flowschemas/status
prioritylevelconfigurations/status
ingresses/status
networkpolicies/status
poddisruptionbudgets/status
volumeattachments/status
```

Встречаем не существующие resources

```
C:\Users\d.evdokimov\Documents>kubectl.exe --kubeconfig=config127 get clusterroles system:certificates.k8s.io:legacy-unknown-approver -ojson
{
  "apiVersion": "rbac.authorization.k8s.io/v1",
  "kind": "ClusterRole",
  "metadata": {
    "annotations": {
      "rbac.authorization.kubernetes.io/autoupdate": "true"
    },
    "creationTimestamp": "2023-05-26T12:00:28Z",
    "labels": {
      "kubernetes.io/bootstrapping": "rbac-defaults"
    },
    "name": "system:certificates.k8s.io:legacy-unknown-approver",
    "resourceVersion": "147",
    "uid": "3c475c99-7f19-447a-8660-726e6c2a497f"
  },
  "rules": [
    {
      "apiGroups": [
        "certificates.k8s.io"
      ],
      "resourceNames": [
        "kubernetes.io/legacy-unknown"
      ],
      "resources": [
        "signers"
      ],
      "verbs": [
        "approve"
      ]
    }
  ]
}
```

Момент 6: много типов взаимоотношении

????



Момент 7: Aggregated ClusterRoles

Есть 3 встроенные роли (на момент 1.27.2):

- admin
- edit
- view

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring
aggregationRule:
  clusterRoleSelectors:
  - matchLabels:
      kubernetes.io/bootstrapping: "rbac-defaults"
rules: [] # The control plane automatically fills in the rules
```

```
C:\Users\d.evdokimov\Documents>kubectl.exe --kubeconfig=config_RBAC describe clusterrole monitoring
```

Name:	monitoring		
Labels:	<none>		
Annotations:	<none>		
PolicyRule:			
Resources	Non-Resource URLs	Resource Names	Verbs
-----	-----	-----	-----
.	[*]	[*]	[* delete get list patch update watch deletecollection]
nodes/log	[*]	[*]	[*]
nodes/metrics	[*]	[*]	[*]
nodes/proxy	[*]	[*]	[*]
nodes/stats	[*]	[*]	[*]
signers.certificates.k8s.io	[*]	[kubernetes.io/kube-apiserver-client-kubelet]	[approve sign]
signers.certificates.k8s.io	[*]	[kubernetes.io/kube-apiserver-client]	[approve sign]
signers.certificates.k8s.io	[*]	[kubernetes.io/kubelet-serving]	[approve sign]
signers.certificates.k8s.io	[*]	[kubernetes.io/legacy-unknown]	[approve sign]
leases.coordination.k8s.io	[*]	[*]	[create delete deletecollection get list patch update watch]
rolebindings.rbac.authorization.k8s.io	[*]	[*]	[create delete deletecollection get list patch update watch]
roles.rbac.authorization.k8s.io	[*]	[*]	[create delete deletecollection get list patch update watch]
configmaps	[*]	[*]	[create delete deletecollection patch update get list watch]
events	[*]	[*]	[create delete deletecollection patch update get list watch]
persistentvolumeclaims	[*]	[*]	[create delete deletecollection patch update get list watch]
Pods	[*]	[*]	[create delete deletecollection patch update get list watch]
replicationcontrollers/scale	[*]	[*]	[create delete deletecollection patch update get list watch]
replicationcontrollers	[*]	[*]	[create delete deletecollection patch update get list watch]

- Политики Kyverno для работы с сущностями RBAC <https://kyverno.io/policies/?policytypes=RBAC> (11 штук)
- Наша политика для контроля ролей с aggregationRule:

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: disallow-clusterrole-aggregationrule
spec:
  validationFailureAction: Enforce
  rules:
    - name: check-clusterrole-aggregationrule
      match:
        resources:
          kinds:
            - ClusterRole
      validate:
        message: "Creation of ClusterRoles with aggregationRule is not allowed."
        pattern:
          X(aggregationRule): "null"
```

Момент 8: Свой RBAC поверх стандартного

["Least-Privilege Kubernetes Authorization with OPA"](#) - Charlie Cetin, Daniel Popescu, & Quentin Long

Least-Privilege Kubernetes Authorization with OPA

Charlie Cetin
Daniel Popescu
Quentin Long

Рекомендации и выводы

- Всегда изучайте что ставите в кластер и с какими правами
- Использование Kubernetes Audit Log
- Использование политик Policy Engine для ресурсов RBAC
- Используйте мощь и гибкость Kubernetes в свою сторону

- RBAC Kubernetes простой только с виду и имеет много не очевидных моментов
- Из-за проблем с RBAC нанести вред системе куда проще чем через уязвимость сервиса
- Контролировать RBAC необходимо постоянно/непрерывно
- Благодаря декларативной природе Kubernetes и расширению через Kubernetes operators можно сделать более гибкую систему авторизации

- [“RBAC to the Future: Untangling Authorization in Kubernetes”](#) - Jimmy Mesta, KubeCon+CloudNativeCon 2023
- [“Cracking Kubernetes RBAC Authorization \(AuthZ\) Model \(2022\)”](#) - Arthur Chiao, Blog
- [“Assign permissions to an user in Kubernetes. An overview of RBAC-based AuthZ in k8s”](#) - Federico Carbonetti
- [“Kubernetes Privilege Escalation: Excessive Permissions in Popular Platforms”](#) - Yuval Avraham, Shaul Ben Hai
- [“Role Based Access Control Good Practices”](#) – Kubernetes documentation
- [“RBAC”](#) – unofficial Kubernetes documentation
- [“Kubernetes RBAC 101”](#) - Oleg Chunikhin
- [rbac.dev](#) – github rep

7 июня 2023 📍 Москва, МЦК ЗИЛ
Первая в России конференция
по БЕзопасности КОНтейнеров и контейнерных сред

БЕΙΚОИЧ



Contacts:

Email: de@luntry.ru

Twitter: [@evdokimovds](https://twitter.com/evdokimovds)

Tg: [@Qu3b3c](https://t.me/@Qu3b3c) [@k8security](https://t.me/@k8security)

Site: www.luntry.ru