揭秘 Kubernetes RBAC 安全：风险、挑战及防护

李昌昊　韦伟　　字节跳动

2024.08.24

RBAC Buster

April 2023 - Aqua Sec 首次借助蜜罐捕获到利用 Kubernetes 的 RBAC 配置风险进行攻击的行为，黑客利用 RBAC 风险创建后门，部署挖矿程序，并试图横向移动到云平台。

SYS:ALL

Jan 2024 - Orca 报告称，通过对暴露的 GKE 集群进行扫描，发现其中 1300 个集群存在错误配置的角色绑定，其中有 108 个集群允许攻击者使用任意 Google 账户接管集群。

OWASP K8s Top 10

在 OWASP Kubernetes Top 10 安全风险中，RBAC 配置错误导致的“权限过多”问题排名第三，可能引发未授权访问和权限提升等问题。

目录
CONTENT

01 K8s AuthN & AuthZ
BACKGROUND

02 揭秘 RBAC 安全现状
THE CURRENT STATE OF RBAC SECURITY

03 安全风险与挑战剖析
RBAC SECURITY RISK AND CHALLENGES

04 纵深防御策略与实践
DEFENSE-IN-DEPTH STRATEGIES AND PRACTICES

PART ONE

01

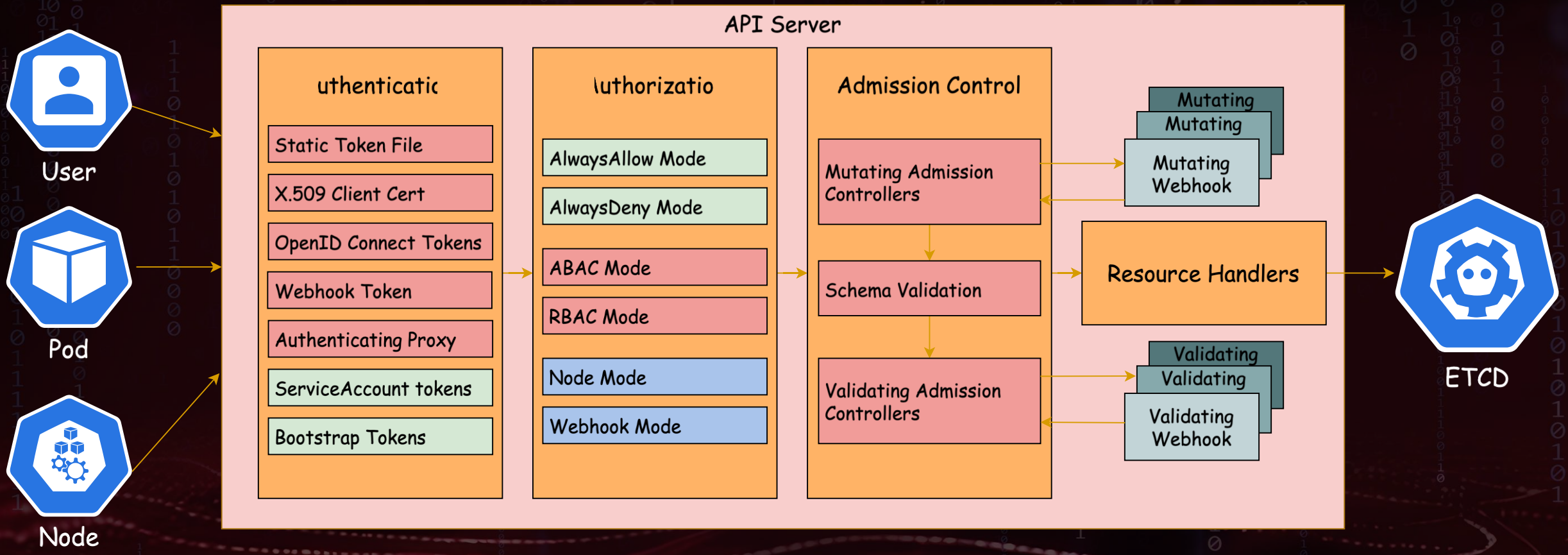
K8s AuthN & AuthZ

The introduction of K8s AuthN/AuthZ

Kubernetes 认证与授权体系——API Server

Kubernetes Authentication and Authorization System – API Server

KCon



Kubernetes 认证与授权体系——Kubelet Server

Kubernetes Authentication and Authorization System – Kubelet Server

认证

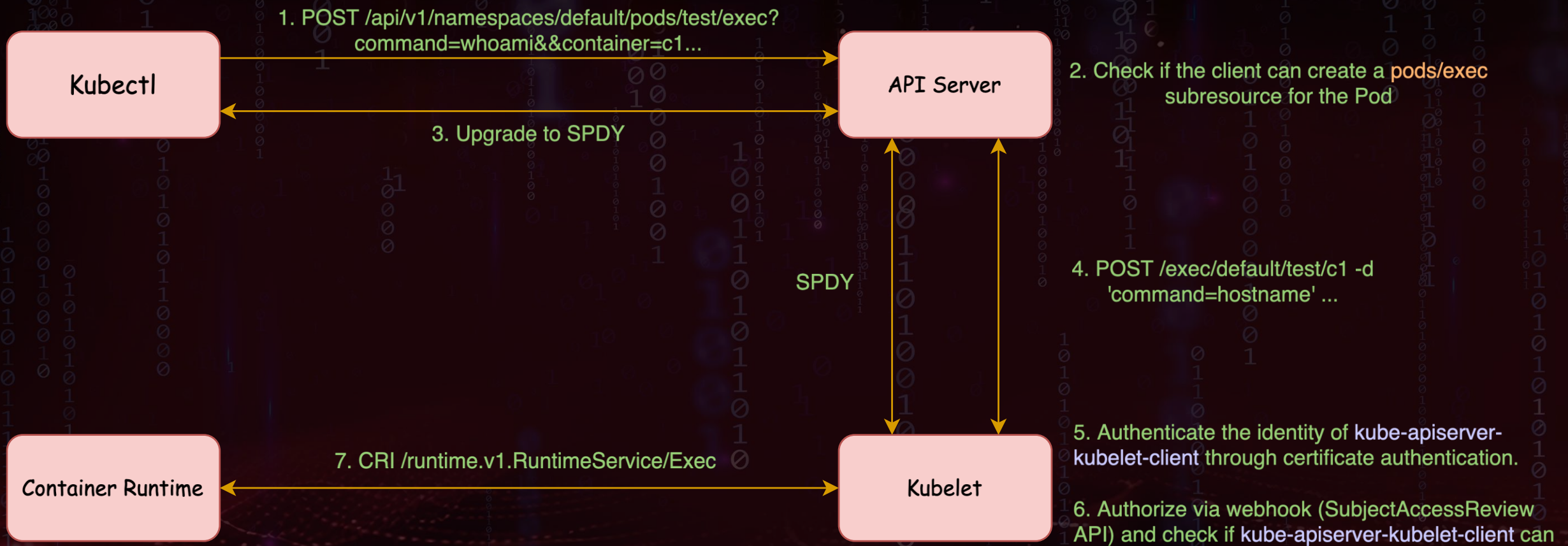
- 匿名认证
- 证书认证：通过 CA 证书对客户端身份进行认证
- Webhook 认证：通过 TokenReview API 向 API Server 发起身份认证请求

授权

- AlwaysAllow 授权：任何通过认证（包括匿名用户）的用户都可以得到授权
- Webhook 授权：通过 SubjectAccessReview API 向 API Server 发起权限校验请求

Kubernetes 认证与授权体系——Kubelet Server

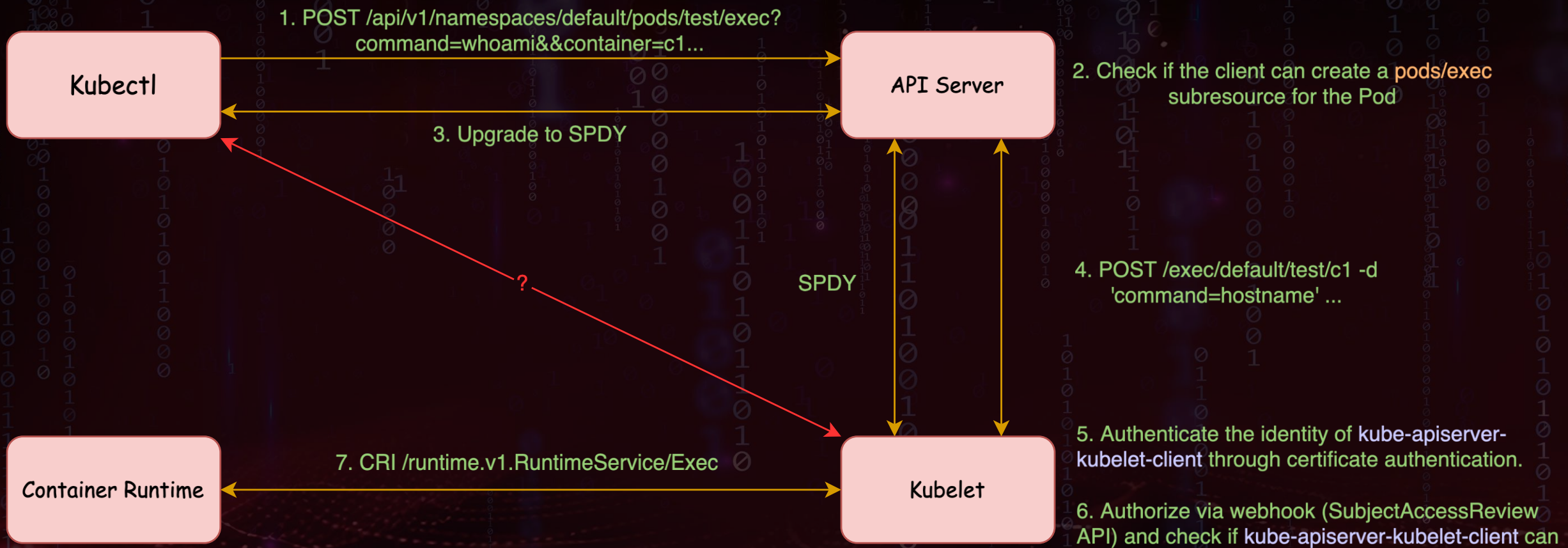
Kubernetes Authentication and Authorization System – Kubelet Server



执行 kubectl exec 后都发生了什么?

Kubernetes 认证与授权体系——Kubelet Server

Kubernetes Authentication and Authorization System – Kubelet Server



直接访问 Kubelet Server 有什么风险?

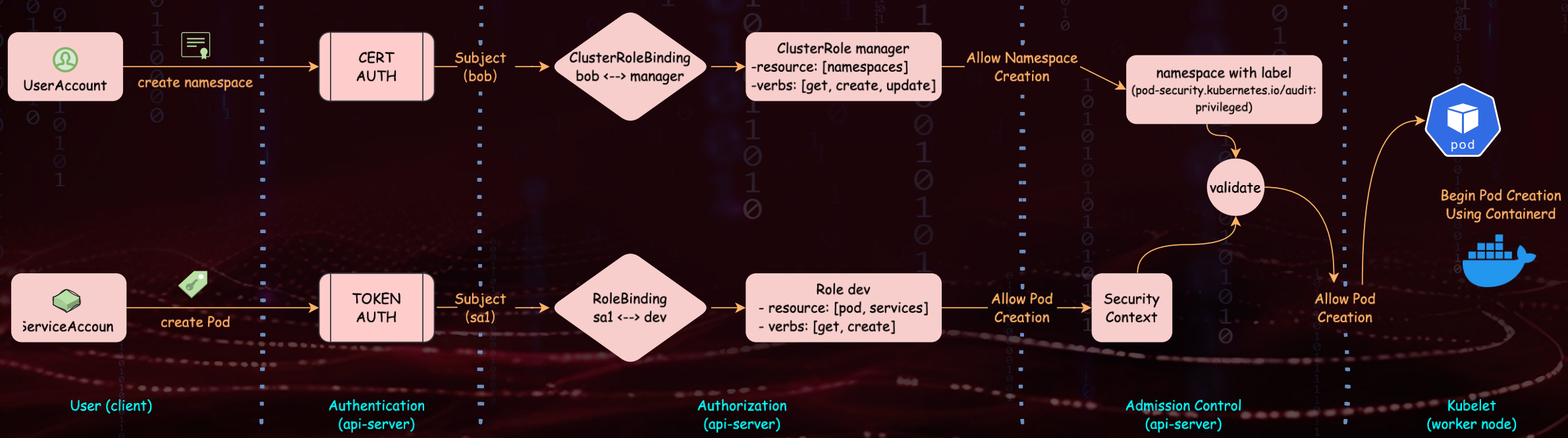
Kubernetes RBAC 授权机制

Kubernetes RBAC Authorization Mechanism

RBAC 是基于角色的访问控制，用于 Kubernetes 集群中资源的授权管理。

关键概念包括：

- 用户账户 UserAccount、服务账户 ServerAccount
- 角色 Role、集群角色 ClusterRole、
- 角色绑定 RoleBinding、集群角色绑定 ClusterRoleBinding
- Deny by Default 安全模型



KCon

PART TWO

02

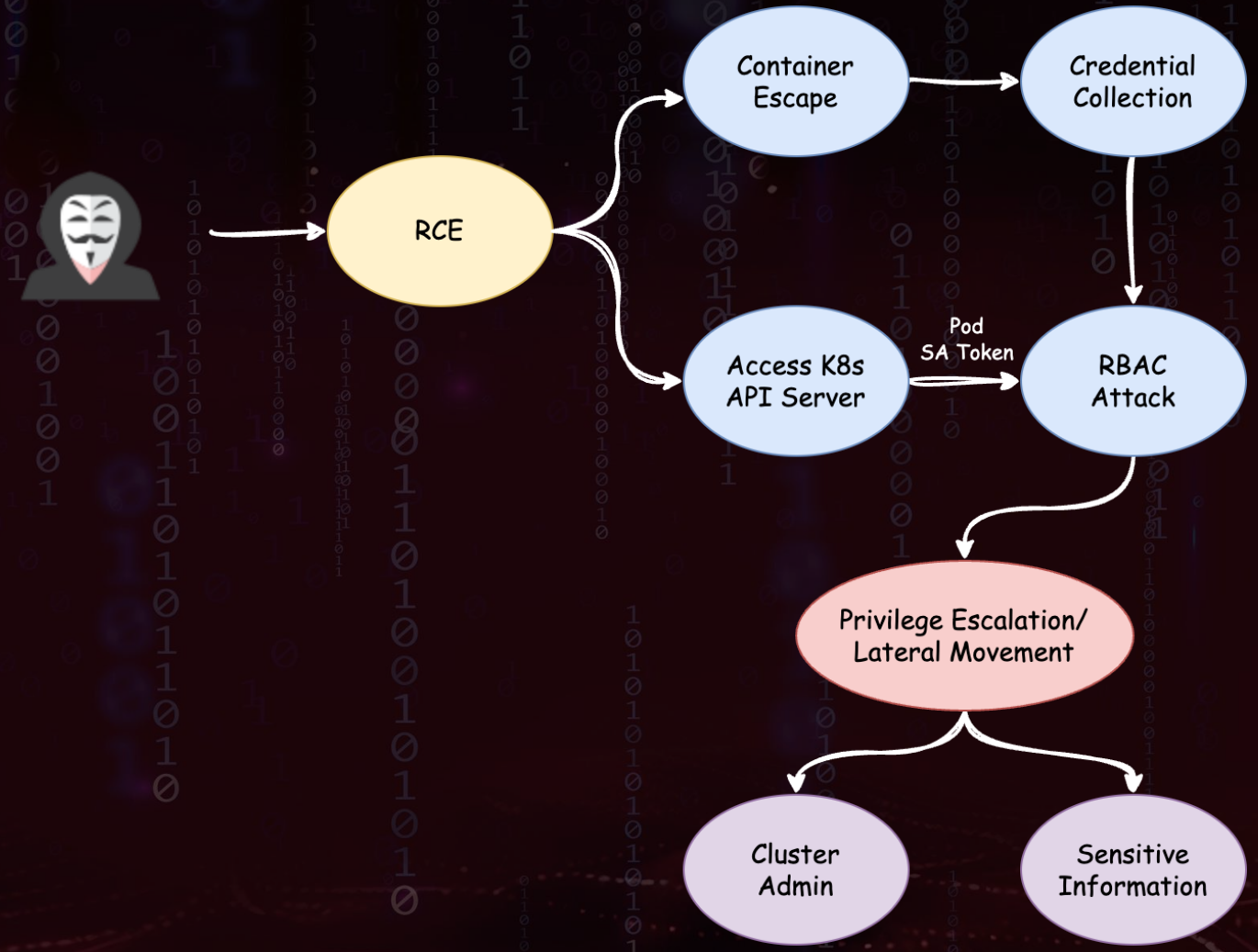
揭秘 RBAC 安全现状

Unveiling the Current State of RBAC Security

典型攻击链路

Typical Attack Vectors in K8s RBAC

- 在云厂商、PaaS 平台、云原生应用、SaaS 产品中呈上升趋势
- 轻则被用于后渗透入侵，重则给用户、产品引入安全漏洞
- 给安全防护工作带来了新的挑战



典型的 RBAC 攻击链路

Kubernetes 中的敏感权限

Sensitive Permissions in Kubernetes

KCon

Manipulate AuthN/Z	Acquire Tokens	RCE	Steal Pods	MitM
impersonate	list secrets	create pods/exec	modify nodes	control endpointslices
escalate	create secrets	update pods/ephemeral-containers	modify nodes/status	modify endpoints
bind	create serviceaccounts/token	create nodes/proxy	create pods/eviction	modify services/status
approve signers	create pods	control pods	delete pods	modify services/status
update certificatesignin-grequests/approval	control pod controllers	control pod controllers	delete nodes	modify pods
control mutating webhooks	control validating webhooks	control mutating webhooks	modify pods/status	create services
—	control mutating webhooks	—	modify pods	control mutating webhooks

引用自《Kubernetes Privilege Escalation: Excessive Permissions in Popular Platforms》

典型攻击手法

Typical Attack Techniques In K8s RBAC

KCon

敏感权限	攻击手法
list or get secrets	窃取敏感 Secret，比如 SA Token、数据库凭证等
escalate roles/clusterroles	通过修改角色权限，为其授予特权来进行提权
bind roles/clusterroles	创建或修改 rolebindbings/clusterrolebindings 来分配不当权限
impersonate subjects	扮演任意用户身份执行操作，进行提权或身份伪装
approve signers	通过 CSR API 签发证书，伪造任意用户身份
create serviceaccounts/token	通过 TokenRequest API 创建并获取指定 SA 的 Token
create nodes/proxy	直接访问 Kubelet Server，在容器内执行命令
create workloads	创建恶意工作负载进行提权，比如特权容器、窃取指定 SA token
create PV & PVC	创建并绑定恶意的持久化存储卷，访问节点上的所有数据

风险举例——为特权服务账户颁发 token

Issuing Tokens for Privileged Service Accounts

威胁 RBAC 资源：

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: issue-token-secrets
rules:
# 可通过 create secret 来窃取指定的 SA token
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["create", "get"]

# 可通过 patch secret 来窃取指定的 SA token
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["patch", "get"]
```

注： 自 v1.24 起，K8s 不再为 SA 创建 Secret 对象。因此无法利用 patch 权限窃取其他 SA 的 token

风险举例——利用 get secrets 爆破提权

Privilege Escalation through get secrets Exploitation

威胁 RBAC 资源：

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

SA 的 Secret 命名格式为：固定前缀 + 五个随机字符（由一个有限的 27 个字符集生成），总计 14348907 种组合（27^5）。因此，攻击者可以在数小时内通过暴力破解推导出 Secret，从而访问敏感服务帐户并提升权限。

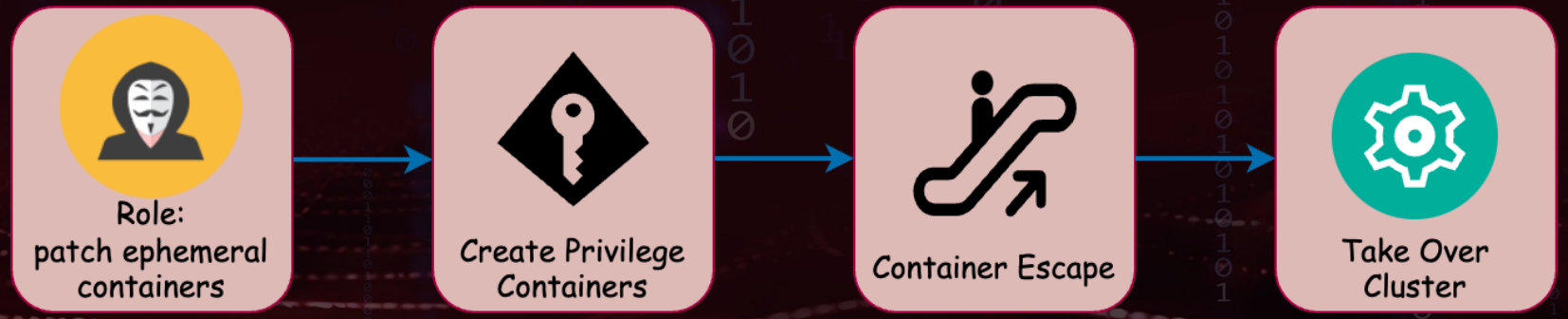
风险举例——利用 ephemeralContainers 提权

Privilege Escalation using Ephemeral Containers

威胁 RBAC 资源：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: create-ephemeralconatiner
  namespace: test
rules:
- apiGroups: [""]
  resources: ["pods/ephemeralcontainers"]
  verbs: ["patch"]
```

```
curl -k -X PATCH \
-H "Authorization: Bearer {TOKEN}" \
-H "Content-Type: application/strategic-merge-patch+json" \
https://{APISERVER}/api/v1/namespaces/test/pods/test-normal/ephemeralcontainers \
-d
'{"spec":{"ephemeralContainers":[{"name":"debugger","image":"busybox:latest","command":["sh"],"stdin":true,"tty":true,"securityContext":{"privileged":true}}]}}'
```



注：Ephemeral Containers 自 v1.23 GA

风险举例——利用 pods/eviction 提权

Privilege Escalation Using Ephemeral Containers

威胁 RBAC 资源：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: create-ephemeralconatiner
  namespace: test
rules:
- apiGroups: [""]
  resources: ["pods/eviction"]
  verbs: ["create"]
```

```
curl -k -X PATCH \
-H "Authorization: Bearer {TOKEN}" \
-H "Content-Type: application/strategic-merge-patch+json" \
https://{APISERVER}/api/v1/namespaces/ingress-
nginx/pods/ingress-nginx-controller-59669cf454-
9z2xk/eviction \
-d
'{"apiVersion":"policy/v1","kind":"Eviction","metadata":{"name":
" ingress-nginx-controller-59669cf454-
9z2xk","namespace":"  ingress-nginx"}}'
```

尝试驱逐特权组件的 Pod， 将其调度到攻击者所在节点

风险举例——直接访问 Kubelet 躲避检测

Directly Access Kubelet Server to Evade Detection

- Kubelet 使用 SubjectAccessReview API 对客户端进行鉴权
  (例如执行命令时需要 nodes/proxy 资源的 create 权限)
- 对应 Audit Event 仅记录请求者的身份信息
- 可利用 nodes/proxy、nodes/log 等资源权限访问 Kubelet 并躲避检测

Audit Event

```
"spec": {
    "uid": "3538dc71-3058-4b32-8005-04148b3262c4",
    "user": "system:serviceaccount:default:nodeproxy",
    "groups": [
      "system:serviceaccounts",
      "system:serviceaccounts:default",
      "system:authenticated"
    ],
    "resourceAttributes": {
      "resource": "nodes",
      "subresource": "proxy",
      "verb": "create",
      "name": " cn-beijing.172.16.0.187",
      "version": "v1"
    },
},
"status": {
    "reason": "...",
    "allowed": true
}
```

风险举例——为系统用户、用户组授权

Granting Permissions to System Users and User Groups

KCon

典型错误配置：

- system:nodes 用户组被授予额外权限，导致 kubelet 的凭据绕过了“Node Authorizer”的约束。

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: bypass-node-authorizer
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: test-clusterrole   # The role has secrets lists permission
subjects:
- kind: Group
  name: system:nodes
```

```
root@iv-yd8g2vo6wwajrd7g2pvg:~# export KUBECONFIG=/et
c/kubernetes/kubelet.conf
root@iv-yd8g2vo6wwajrd7g2pvg:~# kubectl get secrets -
n kube-system
Error from server (Forbidden): secrets is forbidden:
User "system:node:192.168.0.6" cannot list resource "
secrets" in API group "" in the namespace "kube-syste
m": No Object name found
root@iv-yd8g2vo6wwajrd7g2pvg:~#
```

```
root@iv-yd8g2vo6wwajrd7g2pvg:~# export KUBECONFIG=/et
c/kubernetes/kubelet.conf
root@iv-yd8g2vo6wwajrd7g2pvg:~# kubectl get secrets -
n kube-system
NAME                                                  TYPE
                                DATA   AGE
addon.autoscaler.token                                Opaque
                                1      30d
addon.csi.token                                       Opaque
                                1      30d
addon.network.token                                   Opaque
                                1      30d
```

风险举例——隐蔽的 list secrets 授权

Hidden Risks of list secrets Authorization

在 Kubernetes RBAC 中，如果设置了 resourceNames 字段，那么请求的权限不能是 list、watch、create、deletecollection，否则请求不会被允许。

典型错误配置：

- 当 resourceNames 中存在名为 "" 的资源名称时，将允许 list 请求。

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: hidden-authorization
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["list"]
  resourceNames: ["specialSecret", ""]
```

```
bytedance@FHFW5HKW1H rbac-vuln-test % kubectl auth can-i --list
Resources                                       Non-Resource URLs                      Resource Names   Verbs
selfsubjectaccessreviews.authorization.k8s.io   []                                     []               [create]
selfsubjectrulesreviews.authorization.k8s.io    []                                     []               [create]
                                                [/.well-known/openid-configuration]    []               [get]
                                                [/api/*]                               []               [get]
                                                [/api]                                 []               [get]
                                                [/apis/*]                              []               [get]
                                                [/apis]                                []               [get]
                                                [/healthz]                             []               [get]
                                                [/healthz]                             []               [get]
                                                [/livez]                               []               [get]
                                                [/livez]                               []               [get]
                                                [/openapi/*]                           []               [get]
                                                [/openapi]                             []               [get]
                                                [/openid/v1/jwks]                      []               [get]
                                                [/readyz]                              []               [get]
                                                [/readyz]                              []               [get]
                                                [/version/]                            []               [get]
                                                [/version/]                            []               [get]
                                                [/version]                             []               [get]
                                                [/version]                             []               [get]
secrets                                         []                                     []               [list]
bytedance@FHFW5HKW1H rbac-vuln-test % kubectl get secrets -n kube-system
NAME                                  TYPE     DATA   AGE
addon.autoscaler.token                Opaque   1      48d
addon.csi.token                       Opaque   1      48d
addon.network.token                   Opaque   1      48d
addon.observability.token             Opaque   1      48d
```

KCon

PART THREE

03

RBAC 安全风险与挑战剖析

RBAC Security Risk and Challenges

风险如何被引入

How RBAC Risks are Introduced

开发时未遵循最小权限原则

开发人员在构建应用时，由于安全意识不足，没有按照最小权限原则申请权限，导致风险。同时随着系统的复杂性增加，权限角色的管理也变得愈发复杂，随着版本迭代的过程，也会引入新的风险。

使用存在风险的开源组件

许多开源组件在默认配置下可能授权过于宽泛的权限。这种「开箱即用」的配置虽方便，但可能不符合特定部署的安全需求。

部署组件时未考虑安全风险

不当部署组件容易引入安全风险。优先使用专用命名空间隔离需要敏感权限的控制组件；且不应将控制组件与业务负载混部，以减少潜在的安全威胁。

风险如何被引入——使用存在风险的开源组件

Using Vulnerable Open Source Components

KCon

kube-state-metrics

```
kind: ClusterRole
...
rules:
...
- apiGroups:
  - ""
  resources:
  - configmaps
  - secrets
  - nodes
  - pods
  - services
  - resourcequotas
  - replicationcontrollers
  - limitranges
  - persistentvolumeclaims
  - persistentvolumes
  - namespaces
  - endpoints
  verbs:
  - get
  - list
  - watch
```

⚠ kube-state-metrics 默认监控全部的 secrets，但不是所有场景都需要。

Ingress-nginx

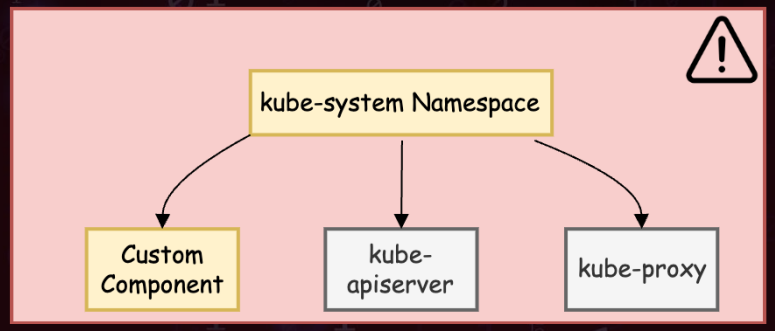
```
kind: ClusterRole
...
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  - endpoints
  - nodes
  - pods
  - secrets
{{- if not .Values.controller.scope.enabled }}
  - namespaces
{{- end}}
  verbs:
  - list
  - watch
```

⚠ Ingress-nginx 需要 secrets 的 list, watch 权限用于管理 Ingress 的 TLS 证书&密钥。

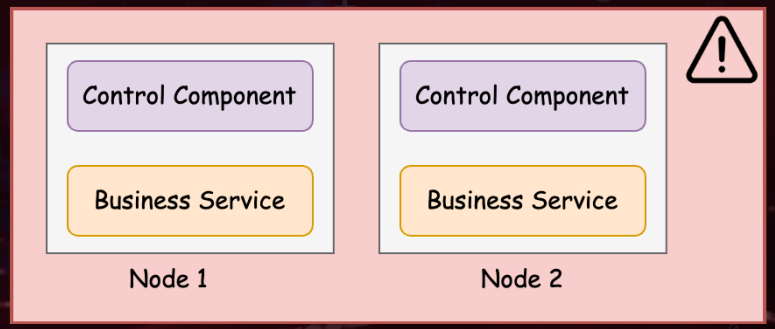
风险如何被引入——部署组件时未考虑安全风险

Ignoring Security Risks During Component Deployment

- 将控制组件部署在 kube-system、default 等命名空间



- 控制组件与业务负载混部在同一个节点



防御 RBAC 安全风险的挑战

Challenges in Defending Against RBAC Security Risks

依赖端上防护

企业生产网络过于依赖端上防护，缺少 Kubernetes 层面的防御手段

RBAC 风险具有隐蔽性

攻击者利用 RBAC 安全风险进行渗透具有隐蔽性

RBAC 风险较大

利用 RBAC 安全风险可迅速扩大战果，控制整个集群

收敛风险难度大

涉及基础设施，可能已经大规模部署
风险可能来自外部组件
缺少必要知识和手段

KCon

PART FOUR

04

纵深防御策略与实践

Defense-in-Depth Strategies and Practices

业务 vs 安全

Business vs. Security

KCon

建设治理所需的能力（知识、工具和系统）

平衡业务发展和安全要求，自顶向下对齐目标

加强反入侵建设，增强兜底能力

根据企业现状制定计划

红蓝演练验证风险

业务 vs 安全

Business vs. Security

建设治理所需的能力（知识、工具和系统）

平衡业务发展和安全要求，自顶向下对齐目标

加强反入侵建设，增强兜底能力

根据企业现状制定计划

红蓝演练验证风险

制定治理计划

Developing a Feasible Governance Plan

- 数据驱动
  持续扫描识别风险、评估严重性、定位风险来源
- 分级治理
  结合风险的可利用性、严重性和影响范围，制定优先级
- 增量管控
- 存量整改

制定治理计划——分级治理

Developing a Feasible Governance Plan – Tiered Governance

KCon

P0 - 不应被组件使用	可直接获取整个集群访问控制权限的权限。 如 cluster admin, approve csrs	如无特殊情况不应使用。
P1 - 不应被组件使用	可间接获取整个集群访问控制权限的权限。 如 impersonate, bind, escalate, list all secrets ...	如无特殊情况不应使用。
P2 - 需严格控制范围	需要一定知识或前置条件，可间接控制整个集群的权限。 如 issue kube-system token secrets, webhook control, assin sa ...	如业务强依赖，需说明合理性并配合其他安全手段进行安全检测、加固。
P3 - 需尽可能收敛	可用于命令执行、横向移动、权限提升等攻击行为的权限。 如 ephemeralContainer, pods/exec, nodes/proxy ...	如业务强依赖，需说明合理性并配合其他安全手段进行安全检测、加固。
P4 - 视情况判断	可用于中间人攻击、工作负载调度等攻击行为的权限。 如 node/status, weak ns serviceaccounts/token || secrets ...	在特定场景中存在安全风险，可在评估后使用，并通过入侵检测手段进行兜底。

业务 vs 安全

Business vs. Security

KCon

建设治理所需的能力（知识、工具和系统）

平衡业务发展和安全要求，自顶向下对齐目标

加强反入侵建设，增强兜底能力

根据企业现状制定计划

红蓝演练验证风险

建设治理能力

Governance Measures of K8s RBAC

KCon

最佳实践

指导研发团队进行安全设计和开发

整改建议

指导研发团队对敏感权限整改

增量管控

CI/CD、PaaS 平台、K8s 集群集成

存量扫描

持续识别、定位账户的 RBAC 权限风险

行为建模

基于账户行为生成最小权限角色，辅助整改

建设治理能力——最佳实践

Governance Measures – RBAC Best Security Practices

KCon

01

Kcon 2024

最小权限原则

遵循最小权限原则并授予执行任务所需的最低权限。

02

Kcon 2024

避免为 default 服务账号授予权限

创建 Pod 如果未指定 SA，会默认将命名空间内名为 default 的 SA 分配给 Pod。

03

Kcon 2024

避免使用默认角色/用户/用户组

许多默认角色都具有较高的权限，默认用户、用户组具有特殊用途。

04

Kcon 2024

尽量避免使用通配符

通配符可能会引入额外的安全风险，建议在 RBAC 规则中指定明确的API 组、资源、动词，甚至是资源名称。

05

Kcon 2024

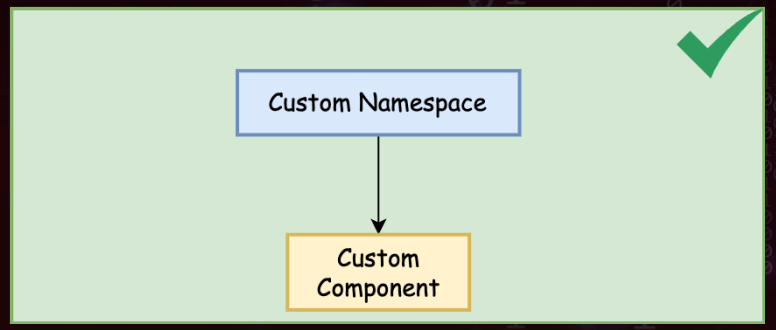
使用单独规则对特定资源授予权限

使用单独规则授予权限，可确保每个主体仅获得所需的最小权限，同时保持规则的独立性和清晰性，从而提高整体管理效率

建设治理能力——安全编排

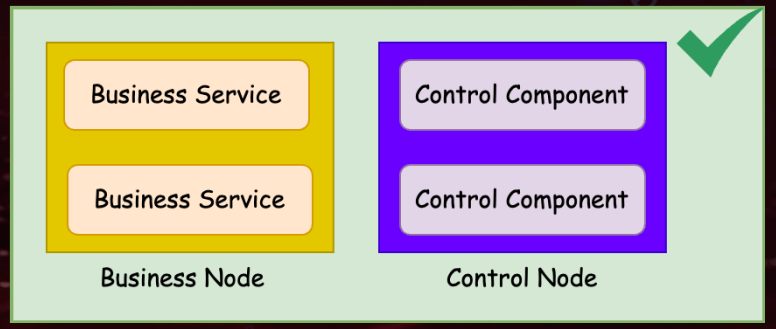
Governance Measures – Security Orchestration

- 控制组件优先考虑部署在专用命名空间



- 将 Role 限定在专用命名空间，收敛权限范围
- 与 default 命名空间、业务负载命名空间隔离，降低凭据泄漏风险

- 控制组件尽量避免与业务负载混部



- 降低节点沦陷后的凭据泄漏风险

建设治理能力——行为建模

Governance Measures – Behavior Modeling

KCon

应按照最小权限原则声明权限

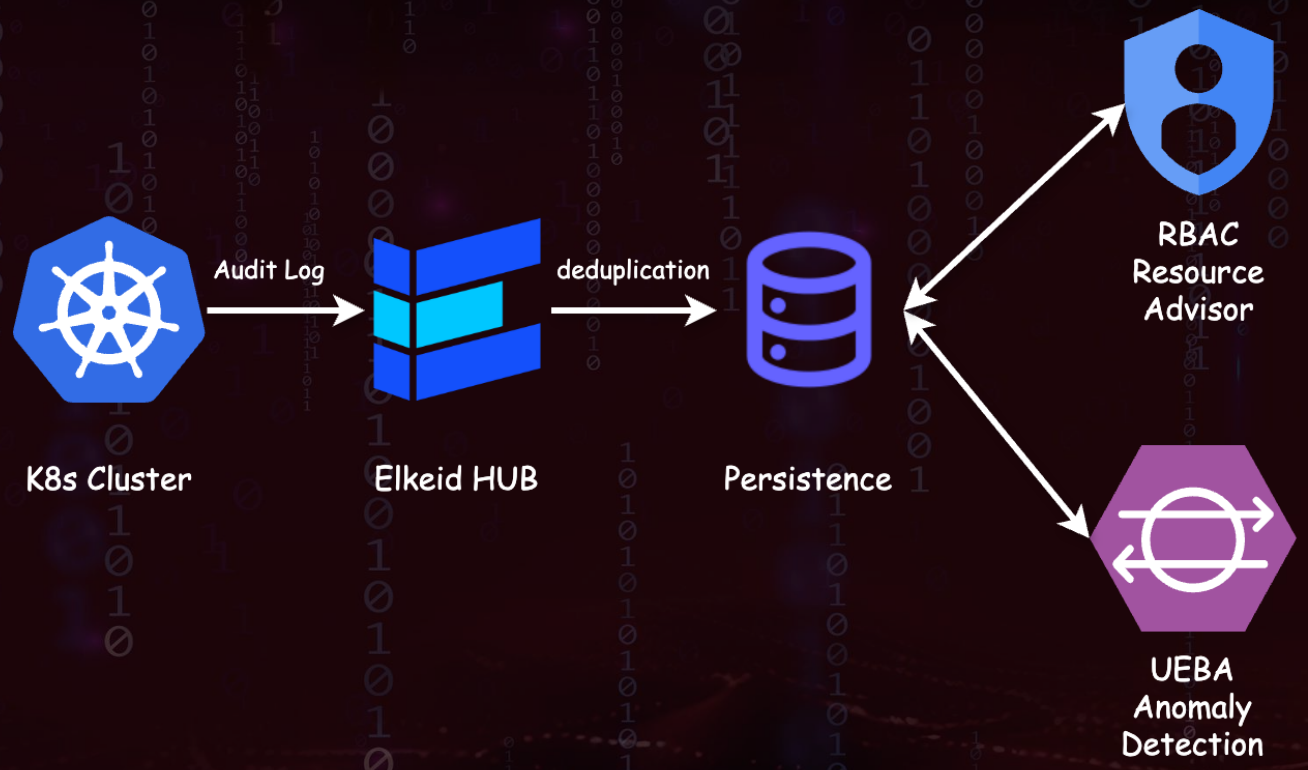
但组件到底需要哪些权限？

审计日志记录了账户的实际访问行为，能否基于审计日志对账户进行权限建模。从而生成角色定义，以指导权限治理？甚至是进行异常检测？

建设治理能力——行为建模

Governance Measures – Behavior Modeling

- 实时处理全量审计日志数据
  - 自动加白
  - 多级缓存去重
- 角色定义生成
  - 资源角色类型判断
  - 集群自定义资源处理
- 多集群建模与聚类（特殊场景）
  - 机器学习聚类
  - 权限模板生成



建设治理能力——行为建模

Governance Measures – Behavior Modeling

- 如何判断集群和命名空间范围资源?
  - 通过 API 获取资源类型
- 如何处理 CRD 资源?
  - 通过集群版本默认的资源进行判断
- 如何尽可能生成最小权限?
  - ClusterRole
  - Role

建设治理能力——行为建模

Governance Measures – Behavior Modeling

KCon

治理前的角色定义

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: metric-apiserver
rules:
- apiGroups: [""]
resources: ["*"]
verbs: ["*"]
```

生成的角色定义

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: metric-apiserver
rules:
- apiGroups: [""]
resources: ["pods", "nodes"]
verbs: ["get", "list", "watch"]
```

业务 vs 安全

Business vs. Security

建设治理所需的能力（知识、工具和系统）

平衡业务发展和安全要求，自顶向下对齐目标

加强反入侵建设，增强兜底能力

根据企业现状制定计划

红蓝演练验证风险

安全治理 & 入侵检测

Security Governance & Intrusion Detection

KCon

- 不是所有的敏感权限都能够得到治理，需结合治理策略 & 入侵检测进行兜底保障
- 安全治理不应被视作“非黑即白”，即便控制组件的某些敏感权限无法收敛，最小化权限仍然对降低风险、增加入侵检测的机率有重要作用。
- Kubernetes 审计日志可以记录集群中的所有活动，为发现潜在的攻击行为提供了重要的手段。

权限探测

对自身、多个账户的权限进行探测

凭据滥用

滥用 ServiceAccount 访问 API Server

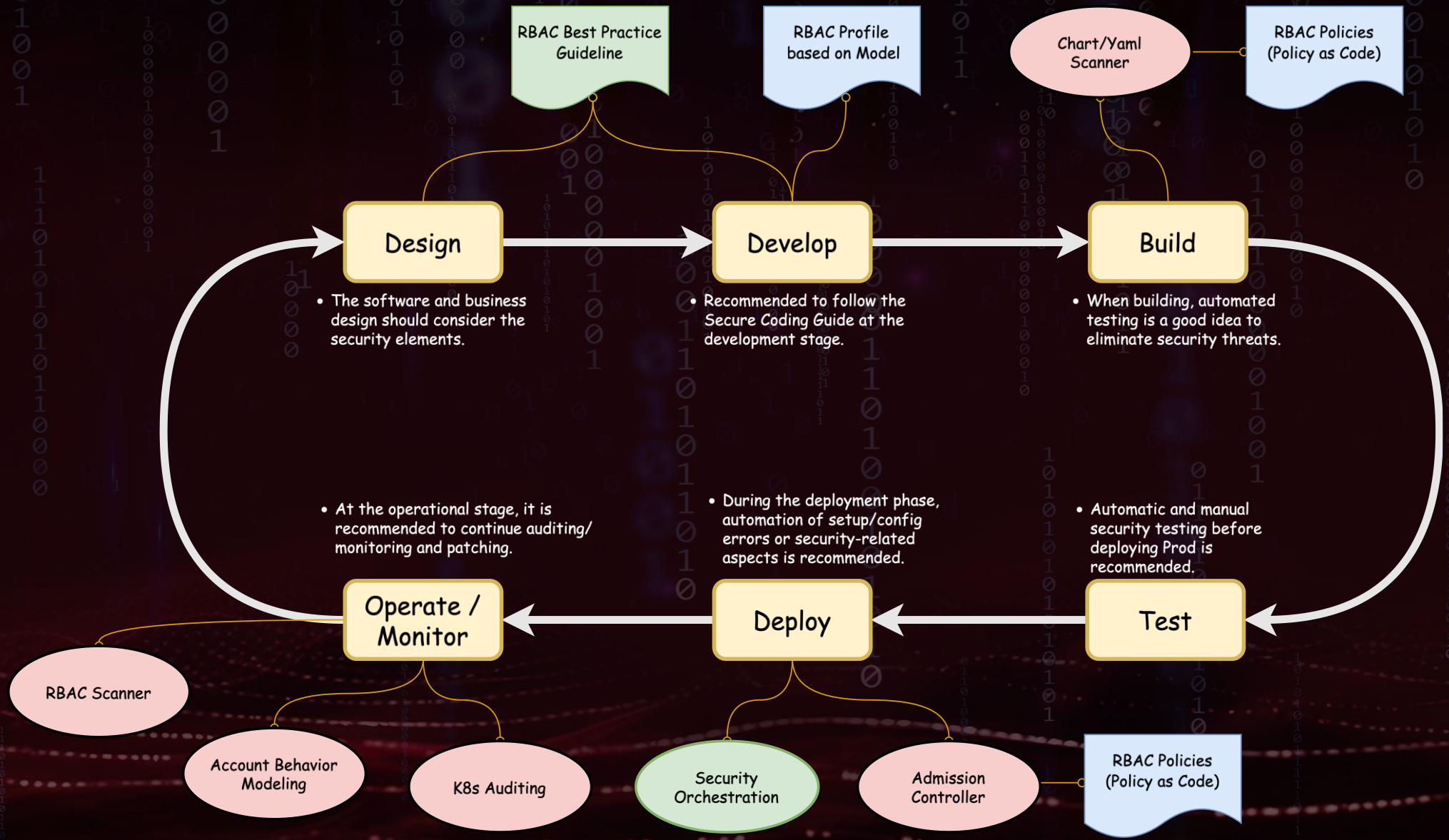
威胁资源

特权容器、特权 Cap 容器、敏感挂载容器、大权限 RBAC 资源等

RBAC 安全防护和治理框架

RBAC Security Protection and Governance Framework

KCon



总结

Takeaways

- 正确管理集群的 RBAC 权限对于保障云原生安全至关重要
- 集群 RBAC 权限治理和管控需平衡安全要求和业务需求，周期长、难度大
- 应持续加强反入侵能力建设，为 RBAC 安全风险提供兜底保障
- 充分挖掘集群审计功能的潜力，能够为安全治理和入侵检测产生许多价值

总结

Takeaways

我们会将《Kubernetes RBAC 最佳安全实践》发布到字节跳动技术团队公众号，敬请期待！

谢谢！