Hardening Guide v2.4

Contents

This document provides prescriptive guidance for hardening a production installation of Rancher v2.4. It outlines the configurations and controls required to address Kubernetes benchmark controls from the Center for Information Security (CIS).

> This hardening guide describes how to secure the nodes in your cluster, and it is recommended to follow this guide before installing Kubernetes.

This hardening guide is intended to be used with specific versions of the CIS Kubernetes Benchmark, Kubernetes, and Rancher:

Hardening Guide Version	Rancher Version	CIS Benchmark Version	Kubernetes Version
Hardening Guide v2.4	Rancher v2.4	Benchmark v1.5	Kubernetes 1.15

Click here to download a PDF version of this document

Overview

This document provides prescriptive guidance for hardening a production installation of Rancher v2.4 with Kubernetes v1.15. It outlines the configurations required to address Kubernetes benchmark controls from the Center for Information Security (CIS).

For more detail about evaluating a hardened cluster against the official CIS benchmark, refer to the **CIS Benchmark Rancher Self-Assessment Guide - Rancher v2.4**.

Known Issues

- **Rancher exec shell and view logs for pods are not functional in a CIS 1.5 hardened setup when only public IP is provided when registering custom nodes. This functionality requires a private IP to be provided when registering the custom nodes.**
- **When setting the `default_pod_security_policy_template_id:` to `restricted` Rancher creates RoleBindings and ClusterRoleBindings on the default service accounts. The CIS 1.5 5.1.5 check requires the default service accounts have no roles or cluster roles bound to it apart from the defaults. In addition the default service accounts**

should be configured such that it does not provide a service account token and does not have any explicit rights assignments.

Configure Kernel Runtime Parameters

The following `sysctl` configuration is recommended for all nodes type in the cluster. Set the following parameters in `/etc/sysctl.d/90-kubelet.conf`:

```
vm.overcommit_memory=1
vm.panic_on_oom=0
kernel.panic=10
kernel.panic_on_oops=1
kernel.keys.root_maxbytes=25000000
```

Run `sysctl -p /etc/sysctl.d/90-kubelet.conf` to enable the settings.

Configure `etcd` user and group

A user account and group for the **etcd** service is required to be setup prior to installing RKE. The **uid** and **gid** for the **etcd** user will be used in the RKE **config.yml** to set the proper permissions for files and directories during installation time.

create `etcd` user and group

To create the **etcd** group run the following console commands.

```
groupadd --gid 52034 etcd
useradd --comment "etcd service account" --uid 52034 --gid
52034 etcd
```

Update the RKE **config.yml** with the **uid** and **gid** of the **etcd** user:

```
services:
  etcd:
    gid: 52034
    uid: 52034
```

Set `automountServiceAccountToken` to `false` for `default` service accounts

Kubernetes provides a default service account which is used by cluster workloads where no specific service account is assigned to the pod. Where access to the Kubernetes API from a pod is required, a specific service account should be created for that pod, and rights granted to that service account. The default service account should be configured such that it does not provide a service account token and does not have any explicit rights assignments.

For each namespace the **default** service account must include this value:

```
automountServiceAccountToken: false
```

Save the following yaml to a file called `account_update.yaml`

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: default
automountServiceAccountToken: false
```

Create a bash script file called `account_update.sh`. Be sure to `chmod +x account_update.sh` so the script has execute permissions.

```
#!/bin/bash -e

for namespace in $(kubectl get namespaces -A -o json | jq -r
'.items[].metadata.name'); do
  kubectl patch serviceaccount default -n ${namespace} -p "$
(cat account_update.yaml)"
done
```

Ensure that all Namespaces have Network Policies defined

Running different applications on the same Kubernetes cluster creates a risk of one compromised application attacking a neighboring application. Network segmentation is important to ensure that containers can communicate only with those they are supposed to. A

network policy is a specification of how selections of pods are allowed to communicate with each other and other network endpoints.

Network Policies are namespace scoped. When a network policy is introduced to a given namespace, all traffic not allowed by the policy is denied. However, if there are no network policies in a namespace all traffic will be allowed into and out of the pods in that namespace. To enforce network policies, a CNI (container network interface) plugin must be enabled. This guide uses **canal** to provide the policy enforcement. Additional information about CNI providers can be found **here**

Once a CNI provider is enabled on a cluster a default network policy can be applied. For reference purposes a **permissive** example is provide below. If you want to allow all traffic to all pods in a namespace (even if policies are added that cause some pods to be treated as "isolated"), you can create a policy that explicitly allows all traffic in that namespace. Save the following `yaml` as `default-allow-all.yaml`. Additional **documentation** about network policies can be found on the Kubernetes site.

> This `NetworkPolicy` is not recommended for production use

```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-allow-all
spec:
  podSelector: {}
  ingress:
  - {}
  egress:
  - {}
  policyTypes:
  - Ingress
  - Egress
```

Create a bash script file called `apply_networkPolicy_to_all_ns.sh`. Be sure to `chmod +x apply_networkPolicy_to_all_ns.sh` so the script has execute permissions.

```
#!/bin/bash -e

for namespace in $(kubectl get namespaces -A -o json | jq -r
'.items[].metadata.name'); do
  kubectl apply -f default-allow-all.yaml -n ${namespace}
done
```

Execute this script to apply the `default-allow-all.yaml` the **permissive** `NetworkPolicy` to all namespaces.

Reference Hardened RKE `cluster.yml` configuration

The reference `cluster.yml` is used by the RKE CLI that provides the configuration needed to achieve a hardened install of Rancher Kubernetes Engine (RKE). Install **documentation** is provided with additional details about the configuration items.

```
# If you intend to deploy Kubernetes in an air-gapped
environment,
# please consult the documentation on how to configure custom
RKE images.
kubernetes_version: "v1.15.9-rancher1-1"
enable_network_policy: true
default_pod_security_policy_template_id: "restricted"
services:
  etcd:
    uid: 52034
    gid: 52034
  kube-api:
    pod_security_policy: true
    secrets_encryption_config:
      enabled: true
    audit_log:
      enabled: true
    admission_configuration:
    event_rate_limit:
      enabled: true
  kube-controller:
    extra_args:
```

```
      feature-gates: "RotateKubeletServerCertificate=true"
  scheduler:
    image: ""
    extra_args: {}
    extra_binds: []
    extra_env: []
  kubelet:
    generate_serving_certificate: true
    extra_args:
      feature-gates: "RotateKubeletServerCertificate=true"
      protect-kernel-defaults: "true"
      tls-cipher-suites: "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA
256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH
_CHACHA20_POLY1305,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_E
CDHE_RSA_WITH_CHACHA20_POLY1305,TLS_ECDHE_ECDSA_WITH_AES_256_G
CM_SHA384,TLS_RSA_WITH_AES_256_GCM_SHA384,TLS_RSA_WITH_AES_128
_GCM_SHA256"
    extra_binds: []
    extra_env: []
    cluster_domain: ""
    infra_container_image: ""
    cluster_dns_server: ""
    fail_swap_on: false
  kubeproxy:
    image: ""
    extra_args: {}
    extra_binds: []
    extra_env: []
network:
  plugin: ""
  options: {}
  mtu: 0
  node_selector: {}
authentication:
  strategy: ""
  sans: []
  webhook: null
addons: |
```

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: ingress-nginx
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: default-psp-role
  namespace: ingress-nginx
rules:
- apiGroups:
  - extensions
  resourceNames:
  - default-psp
  resources:
  - podsecuritypolicies
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: default-psp-rolebinding
  namespace: ingress-nginx
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: default-psp-role
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
---
```

```
apiVersion: v1
kind: Namespace
metadata:
  name: cattle-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: default-psp-role
  namespace: cattle-system
rules:
- apiGroups:
  - extensions
  resourceNames:
  - default-psp
  resources:
  - podsecuritypolicies
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: default-psp-rolebinding
  namespace: cattle-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: default-psp-role
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
---
apiVersion: policy/v1beta1
```

```
    kind: PodSecurityPolicy
    metadata:
      name: restricted
    spec:
      requiredDropCapabilities:
      - NET_RAW
      privileged: false
      allowPrivilegeEscalation: false
      defaultAllowPrivilegeEscalation: false
      fsGroup:
        rule: RunAsAny
      runAsUser:
        rule: MustRunAsNonRoot
      seLinux:
        rule: RunAsAny
      supplementalGroups:
        rule: RunAsAny
      volumes:
      - emptyDir
      - secret
      - persistentVolumeClaim
      - downwardAPI
      - configMap
      - projected
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: ClusterRole
    metadata:
      name: psp:restricted
    rules:
    - apiGroups:
      - extensions
      resourceNames:
      - restricted
      resources:
      - podsecuritypolicies
      verbs:
      - use
```

```
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: ClusterRoleBinding
    metadata:
      name: psp:restricted
    roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: ClusterRole
      name: psp:restricted
    subjects:
    - apiGroup: rbac.authorization.k8s.io
      kind: Group
      name: system:serviceaccounts
    - apiGroup: rbac.authorization.k8s.io
      kind: Group
      name: system:authenticated
    ---
    apiVersion: v1
    kind: ServiceAccount
    metadata:
      name: tiller
      namespace: kube-system
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: ClusterRoleBinding
    metadata:
      name: tiller
    roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: ClusterRole
      name: cluster-admin
    subjects:
    - kind: ServiceAccount
      name: tiller
      namespace: kube-system

  addons_include: []
  system_images:
```

```
    etcd: ""
    alpine: ""
    nginx_proxy: ""
    cert_downloader: ""
    kubernetes_services_sidecar: ""
    kubedns: ""
    dnsmasq: ""
    kubedns_sidecar: ""
    kubedns_autoscaler: ""
    coredns: ""
    coredns_autoscaler: ""
    kubernetes: ""
    flannel: ""
    flannel_cni: ""
    calico_node: ""
    calico_cni: ""
    calico_controllers: ""
    calico_ctl: ""
    calico_flexvol: ""
    canal_node: ""
    canal_cni: ""
    canal_flannel: ""
    canal_flexvol: ""
    weave_node: ""
    weave_cni: ""
    pod_infra_container: ""
    ingress: ""
    ingress_backend: ""
    metrics_server: ""
    windows_pod_infra_container: ""
  ssh_key_path: ""
  ssh_cert_path: ""
  ssh_agent_auth: false
  authorization:
    mode: ""
    options: {}
  ignore_docker_version: false
  private_registries: []
```

```
  ingress:
    provider: ""
    options: {}
    node_selector: {}
    extra_args: {}
    dns_policy: ""
    extra_envs: []
    extra_volumes: []
    extra_volume_mounts: []
  cluster_name: ""
  prefix_path: ""
  addon_job_timeout: 0
  bastion_host:
    address: ""
    port: ""
    user: ""
    ssh_key: ""
    ssh_key_path: ""
    ssh_cert: ""
    ssh_cert_path: ""
  monitoring:
    provider: ""
    options: {}
    node_selector: {}
  restore:
    restore: false
    snapshot_name: ""
  dns: null
```

Reference Hardened RKE Template configuration

The reference RKE Template provides the configuration needed to achieve a hardened install of Kubenetes. RKE Templates are used to provision Kubernetes and define Rancher settings. Follow the Rancher documentaion for additional installation and RKE Template details.

```
  #
  # Cluster Config
  #
```

```
default_pod_security_policy_template_id: restricted
docker_root_dir: /var/lib/docker
enable_cluster_alerting: false
enable_cluster_monitoring: false
enable_network_policy: true
#
# Rancher Config
#
rancher_kubernetes_engine_config:
  addon_job_timeout: 30
  addons: |-
    ---
    apiVersion: v1
    kind: Namespace
    metadata:
      name: ingress-nginx
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: Role
    metadata:
      name: default-psp-role
      namespace: ingress-nginx
    rules:
    - apiGroups:
      - extensions
      resourceNames:
      - default-psp
      resources:
      - podsecuritypolicies
      verbs:
      - use
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: RoleBinding
    metadata:
      name: default-psp-rolebinding
      namespace: ingress-nginx
    roleRef:
```

```
    apiGroup: rbac.authorization.k8s.io
    kind: Role
    name: default-psp-role
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
---
apiVersion: v1
kind: Namespace
metadata:
  name: cattle-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: default-psp-role
  namespace: cattle-system
rules:
- apiGroups:
  - extensions
  resourceNames:
  - default-psp
  resources:
  - podsecuritypolicies
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: default-psp-rolebinding
  namespace: cattle-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
```

```
    kind: Role
    name: default-psp-role
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
---
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted
spec:
  requiredDropCapabilities:
  - NET_RAW
  privileged: false
  allowPrivilegeEscalation: false
  defaultAllowPrivilegeEscalation: false
  fsGroup:
    rule: RunAsAny
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - emptyDir
  - secret
  - persistentVolumeClaim
  - downwardAPI
  - configMap
  - projected
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
    metadata:
      name: psp:restricted
    rules:
    - apiGroups:
      - extensions
      resourceNames:
      - restricted
      resources:
      - podsecuritypolicies
      verbs:
      - use
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: ClusterRoleBinding
    metadata:
      name: psp:restricted
    roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: ClusterRole
      name: psp:restricted
    subjects:
    - apiGroup: rbac.authorization.k8s.io
      kind: Group
      name: system:serviceaccounts
    - apiGroup: rbac.authorization.k8s.io
      kind: Group
      name: system:authenticated
    ---
    apiVersion: v1
    kind: ServiceAccount
    metadata:
      name: tiller
      namespace: kube-system
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: ClusterRoleBinding
    metadata:
      name: tiller
```

```
    roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: ClusterRole
      name: cluster-admin
    subjects:
    - kind: ServiceAccount
      name: tiller
      namespace: kube-system
  ignore_docker_version: true
  kubernetes_version: v1.15.9-rancher1-1
#
#   If you are using calico on AWS
#
#    network:
#      plugin: calico
#      calico_network_provider:
#        cloud_provider: aws
#
# # To specify flannel interface
#
#    network:
#      plugin: flannel
#      flannel_network_provider:
#      iface: eth1
#
# # To specify flannel interface for canal plugin
#
#    network:
#      plugin: canal
#      canal_network_provider:
#        iface: eth1
#
  network:
    mtu: 0
    plugin: canal
#
#    services:
#      kube-api:
```

```
#        service_cluster_ip_range: 10.43.0.0/16
#      kube-controller:
#        cluster_cidr: 10.42.0.0/16
#        service_cluster_ip_range: 10.43.0.0/16
#      kubelet:
#        cluster_domain: cluster.local
#        cluster_dns_server: 10.43.0.10
#
  services:
    etcd:
      backup_config:
        enabled: false
        interval_hours: 12
        retention: 6
        safe_timestamp: false
      creation: 12h
      extra_args:
        election-timeout: '5000'
        heartbeat-interval: '500'
      gid: 52034
      retention: 72h
      snapshot: false
      uid: 52034
    kube_api:
      always_pull_images: false
      audit_log:
        enabled: true
      event_rate_limit:
        enabled: true
      pod_security_policy: true
      secrets_encryption_config:
        enabled: true
      service_node_port_range: 30000-32767
    kube_controller:
      extra_args:
        address: 127.0.0.1
        feature-gates: RotateKubeletServerCertificate=true
        profiling: 'false'
```

```
        terminated-pod-gc-threshold: '1000'
    kubelet:
      extra_args:
        anonymous-auth: 'false'
        event-qps: '0'
        feature-gates: RotateKubeletServerCertificate=true
        make-iptables-util-chains: 'true'
        protect-kernel-defaults: 'true'
        streaming-connection-idle-timeout: 1800s
        tls-cipher-suites: >-

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES
_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305,TLS_ECD
HE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_CHACHA20_POL
Y1305,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_RSA_WITH_AES
_256_GCM_SHA384,TLS_RSA_WITH_AES_128_GCM_SHA256
      fail_swap_on: false
      generate_serving_certificate: true
    scheduler:
      extra_args:
        address: 127.0.0.1
        profiling: 'false'
  ssh_agent_auth: false
windows_prefered_cluster: false
```

Hardened Reference Ubuntu 18.04 LTS cloud-config:

The reference **cloud-config** is generally used in cloud infrastructure environments to allow for configuration management of compute instances. The reference config configures Ubuntu operating system level settings needed before installing kubernetes.

```
#cloud-config
packages:
  - curl
  - jq
runcmd:
  - sysctl -w vm.overcommit_memory=1
  - sysctl -w kernel.panic=10
```

```
  - sysctl -w kernel.panic_on_oops=1
  - curl https://releases.rancher.com/install-docker/18.09.sh
| sh
  - usermod -aG docker ubuntu
  - return=1; while [ $return != 0 ]; do sleep 2; docker ps;
return=$?; done
  - addgroup --gid 52034 etcd
  - useradd --comment "etcd service account" --uid 52034 --
gid 52034 etcd
write_files:
  - path: /etc/sysctl.d/kubelet.conf
    owner: root:root
    permissions: "0644"
    content: |
      vm.overcommit_memory=1
      kernel.panic=10
      kernel.panic_on_oops=1
```