RKE Hardening Guide with CIS v1.20 Benchmark

Contents

This document provides prescriptive guidance for hardening a production installation of a RKE cluster to be used with Rancher v2.6. 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 for RKE clusters and associated with specific versions of the CIS Kubernetes Benchmark, Kubernetes, and Rancher:

Rancher Version	CIS Benchmark Version	Kubernetes Version
Rancher v2.6	Benchmark v1.20	Kubernetes v1.19 up to v1.21

Click here to download a PDF version of this document.

Overview

This document provides prescriptive guidance for hardening a RKE cluster to be used for installing Rancher v2.6 with Kubernetes v1.19 up to v1.21 or provisioning a RKE cluster with Kubernetes v1.19 up to v1.21 to be used within Rancher v2.6. It outlines the configurations required to address Kubernetes benchmark controls from the Center for Information Security (CIS).

For more details about evaluating a hardened cluster against the official CIS benchmark, refer to the CIS 1.20 Benchmark - Self-Assessment Guide - Rancher v2.6.

Known Issues

- Rancher exec shell and view logs for pods are not functional in a CIS v1.20 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` or `restricted-noroot`, based on the pod security policies (PSP) provided by Rancher, Rancher creates RoleBindings and ClusterRoleBindings on the default service accounts. The CIS v1.20 check 5.1.5 requires that 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 before 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 user and group run the following console commands. The commands below use `52034` for uid and gid are for

example purposes. Any valid unused uid or gid could also be used in lieu of `52034`.

```
groupadd --gid 52034 etcd
useradd --comment "etcd service account" --uid 52034 --gid
52034 etcd --shell /usr/sbin/nologin
```

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

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

Configure `default` Service Account

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 including default and kube-system on a standard RKE install, the default service account must include this value:

```
automountServiceAccountToken: false
```

Save the following configuration 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=jsonpath="{.items[*]['metadata.name']}"); do
  kubectl patch serviceaccount default -n ${namespace} -p "$(cat account_update.yaml)"
done
```

Configure Network Policy

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 provided 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 configuration as `default-allow-all.yaml`. Additional documentation about network policies can be found on the Kubernetes site.

> This `NetworkPolicy` is just an example and 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=jsonpath="{.it
ems[*]['metadata.name']}"); do
  kubectl apply -f default-allow-all.yaml -n ${namespace}
done
```

Execute this script to apply the `default-allow-all.yaml` configuration with 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). RKE install documentation is provided with additional details about the configuration items. This reference `cluster.yml` does not include the required nodes directive which will

vary depending on your environment. Documentation for node configuration in RKE can be found here.

```
# If you intend to deploy Kubernetes in an air-gapped environment,
# please consult the documentation on how to configure custom RKE images.
# https://rancher.com/docs/rke/latest/en/installation/ .

# The nodes directive is required and will vary depending on your environment.
# Documentation for node configuration can be found here:
# https://rancher.com/docs/rke/latest/en/config-options/nodes/
nodes: []
services:
  etcd:
    image: ""
    extra_args: {}
    extra_binds: []
    extra_env: []
    win_extra_args: {}
    win_extra_binds: []
    win_extra_env: []
    external_urls: []
    ca_cert: ""
    cert: ""
    key: ""
    path: ""
    uid: 52034
    gid: 52034
    snapshot: false
    retention: ""
    creation: ""
    backup_config: null
  kube-api:
    image: ""
    extra_args: {}
    extra_binds: []
    extra_env: []
```

```
    win_extra_args: {}
    win_extra_binds: []
    win_extra_env: []
    service_cluster_ip_range: ""
    service_node_port_range: ""
    pod_security_policy: true
    always_pull_images: false
    secrets_encryption_config:
      enabled: true
      custom_config: null
    audit_log:
      enabled: true
      configuration: null
    admission_configuration: null
    event_rate_limit:
      enabled: true
      configuration: null
  kube-controller:
    image: ""
    extra_args:
      feature-gates: RotateKubeletServerCertificate=true
      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
      bind-address: 127.0.0.1
    extra_binds: []
    extra_env: []
    win_extra_args: {}
    win_extra_binds: []
    win_extra_env: []
    cluster_cidr: ""
    service_cluster_ip_range: ""
  scheduler:
    image: ""
    extra_args:
```

```
      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
      bind-address: 127.0.0.1
    extra_binds: []
    extra_env: []
    win_extra_args: {}
    win_extra_binds: []
    win_extra_env: []
  kubelet:
    image: ""
    extra_args:
      feature-gates: RotateKubeletServerCertificate=true
      protect-kernel-defaults: true
      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
    extra_binds: []
    extra_env: []
    win_extra_args: {}
    win_extra_binds: []
    win_extra_env: []
    cluster_domain: cluster.local
    infra_container_image: ""
    cluster_dns_server: ""
    fail_swap_on: false
    generate_serving_certificate: true
  kubeproxy:
    image: ""
    extra_args: {}
    extra_binds: []
    extra_env: []
```

```
    win_extra_args: {}
    win_extra_binds: []
    win_extra_env: []
network:
  plugin: ""
  options: {}
  mtu: 0
  node_selector: {}
  update_strategy: null
authentication:
  strategy: ""
  sans: []
  webhook: null
addons: |
  # Upstream Kubernetes restricted PSP policy
  # https://github.com/kubernetes/website/blob/
564baf15c102412522e9c8fc6ef2b5ff5b6e766c/content/en/examples/
policy/restricted-psp.yaml
  apiVersion: policy/v1beta1
  kind: PodSecurityPolicy
  metadata:
    name: restricted-noroot
  spec:
    privileged: false
    # Required to prevent escalations to root.
    allowPrivilegeEscalation: false
    requiredDropCapabilities:
      - ALL
    # Allow core volume types.
    volumes:
      - 'configMap'
      - 'emptyDir'
      - 'projected'
      - 'secret'
      - 'downwardAPI'
      # Assume that ephemeral CSI drivers & persistentVolumes
set up by the cluster admin are safe to use.
      - 'csi'
```

```
      - 'persistentVolumeClaim'
    hostNetwork: false
    hostIPC: false
    hostPID: false
    runAsUser:
      # Require the container to run without root privileges.
      rule: 'MustRunAsNonRoot'
    seLinux:
      # This policy assumes the nodes are using AppArmor
rather than SELinux.
      rule: 'RunAsAny'
    supplementalGroups:
      rule: 'MustRunAs'
      ranges:
        # Forbid adding the root group.
        - min: 1
          max: 65535
    fsGroup:
      rule: 'MustRunAs'
      ranges:
        # Forbid adding the root group.
        - min: 1
          max: 65535
    readOnlyRootFilesystem: false
  ---
  apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRole
  metadata:
    name: psp:restricted-noroot
  rules:
  - apiGroups:
    - extensions
    resourceNames:
    - restricted-noroot
    resources:
    - podsecuritypolicies
    verbs:
    - use
```

```
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: ClusterRoleBinding
    metadata:
      name: psp:restricted-noroot
    roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: ClusterRole
      name: psp:restricted-noroot
    subjects:
    - apiGroup: rbac.authorization.k8s.io
      kind: Group
      name: system:serviceaccounts
    - apiGroup: rbac.authorization.k8s.io
      kind: Group
      name: system:authenticated
    ---
    apiVersion: networking.k8s.io/v1
    kind: NetworkPolicy
    metadata:
      name: default-allow-all
    spec:
      podSelector: {}
      ingress:
      - {}
      egress:
      - {}
      policyTypes:
      - Ingress
      - Egress
    ---
    apiVersion: v1
    kind: ServiceAccount
    metadata:
      name: default
    automountServiceAccountToken: false
  addons_include: []
  system_images:
```

```
    etcd: ""
    alpine: ""
    nginx_proxy: ""
    cert_downloader: ""
    kubernetes_services_sidecar: ""
    kubedns: ""
    dnsmasq: ""
    kubedns_sidecar: ""
    kubedns_autoscaler: ""
    coredns: ""
    coredns_autoscaler: ""
    nodelocal: ""
    kubernetes: ""
    flannel: ""
    flannel_cni: ""
    calico_node: ""
    calico_cni: ""
    calico_controllers: ""
    calico_ctl: ""
    calico_flexvol: ""
    canal_node: ""
    canal_cni: ""
    canal_controllers: ""
    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
  kubernetes_version: ""
  private_registries: []
  ingress:
    provider: ""
    options: {}
    node_selector: {}
    extra_args: {}
    dns_policy: ""
    extra_envs: []
    extra_volumes: []
    extra_volume_mounts: []
    update_strategy: null
    http_port: 0
    https_port: 0
    network_mode: ""
  cluster_name:
  cloud_provider:
    name: ""
  prefix_path: ""
  win_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: {}
    update_strategy: null
    replicas: null
  restore:
    restore: false
```

```
    snapshot_name: ""
  dns: null
  upgrade_strategy:
    max_unavailable_worker: ""
    max_unavailable_controlplane: ""
    drain: null
    node_drain_input: null
```

Reference Hardened RKE Template Configuration

The reference RKE template provides the configuration needed to achieve a hardened install of Kubernetes. RKE templates are used to provision Kubernetes and define Rancher settings. Follow the Rancher documentation for additional installation and RKE template details.

```
  #
  # Cluster Config
  #
  default_pod_security_policy_template_id: restricted-noroot
  docker_root_dir: /var/lib/docker
  enable_cluster_alerting: false
  enable_cluster_monitoring: false
  enable_network_policy: true
  local_cluster_auth_endpoint:
    enabled: true
  name: ''
  #
  # Rancher Config
  #
  rancher_kubernetes_engine_config:
    addon_job_timeout: 45
    authentication:
      strategy: x509
    dns:
      nodelocal:
        ip_address: ''
        node_selector: null
        update_strategy: {}
    enable_cri_dockerd: false
    ignore_docker_version: true
```

```
#
# # Currently only nginx ingress provider is supported.
# # To disable ingress controller, set `provider: none`
# # To enable ingress on specific nodes, use the
node_selector, eg:
#    provider: nginx
#    node_selector:
#      app: ingress
#
  ingress:
    default_backend: false
    default_ingress_class: true
    http_port: 0
    https_port: 0
    provider: nginx
  kubernetes_version: v1.21.8-rancher1-1
  monitoring:
    provider: metrics-server
    replicas: 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
    options:
      flannel_backend_type: vxlan
    plugin: canal
  rotate_encryption_key: false
#
#    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:
    scheduler:
      extra_args:
        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
      bind-address: 127.0.0.1
    etcd:
      backup_config:
        enabled: true
        interval_hours: 12
        retention: 6
        safe_timestamp: false
        timeout: 300
      creation: 12h
```

```
      extra_args:
        election-timeout: 5000
        heartbeat-interval: 500
      retention: 72h
      snapshot: false
      uid: 52034
      gid: 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:
        feature-gates: RotateKubeletServerCertificate=true
        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
      bind-address: 127.0.0.1
    kubelet:
      extra_args:
        feature-gates: RotateKubeletServerCertificate=true
        protect-kernel-defaults: true
        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
  ssh_agent_auth: false
  upgrade_strategy:
    max_unavailable_controlplane: '1'
    max_unavailable_worker: 10%
windows_prefered_cluster: false
```

Reference Hardened cloud-config Configuration

A cloud-config configuration file is generally used in cloud infrastructure environments to allow for configuration management of compute instances. The reference config configures SUSE Linux Enterprise Server (SLES), openSUSE Leap, Red Hat Enterprise Linux (RHEL) and Ubuntu operating system level settings needed before installing Kubernetes.

Reference Hardened cloud-config for SUSE Linux Enterprise Server 15 (SLES 15) and openSUSE Leap 15

```
#cloud-config
system_info:
  default_user:
    groups:
    - docker
write_files:
- path: "/etc/sysctl.d/90-kubelet.conf"
  owner: root:root
  permissions: '0644'
  content: |
    vm.overcommit_memory=1
    vm.panic_on_oom=0
    kernel.panic=10
    kernel.panic_on_oops=1
    kernel.keys.root_maxbytes=25000000
package_update: true
ssh_pwauth: false
runcmd:
# Docker should already be installed in SLES 15 SP3
- zypper install docker containerd
```

```
- systemctl daemon-reload
- systemctl enable docker.service
- systemctl start --no-block docker.service
- sysctl -p /etc/sysctl.d/90-kubelet.conf
- groupadd --gid 52034 etcd
- useradd --comment "etcd service account" --uid 52034 --gid
52034 etcd --shell /usr/sbin/nologin
```

Reference Hardened cloud-config for Red Hat Enterprise Linux 8 (RHEL 8) and Ubuntu 20.04 LTS

```
#cloud-config
system_info:
  default_user:
    groups:
    - docker
write_files:
- path: "/etc/sysctl.d/90-kubelet.conf"
  owner: root:root
  permissions: '0644'
  content: |
    vm.overcommit_memory=1
    vm.panic_on_oom=0
    kernel.panic=10
    kernel.panic_on_oops=1
    kernel.keys.root_maxbytes=25000000
package_update: true
ssh_pwauth: false
runcmd:
# Install Docker from Rancher's Docker installation scripts -
github.com/rancher/install-docker
- curl https://releases.rancher.com/install-docker/20.10.sh |
sh
- sysctl -p /etc/sysctl.d/90-kubelet.conf
- groupadd --gid 52034 etcd
- useradd --comment "etcd service account" --uid 52034 --gid
52034 etcd --shell /usr/sbin/nologin
```