Red Hat

OpenShift Virtualization Hardening Guide

Red Hat OpenShift Product Management

Rev 1.0 - 2nd of June 2025

Context

This document provides prescriptive instructions for creating a more secure, standard configuration baseline for OpenShift Virtualization. Prior to implementing recommendations from this document, Cluster administrators should ensure the OpenShift platform is properly hardened. Compliance Operator provides guidance on supported profiles for hardening both OpenShift Container Platform (OCP) and Red Hat CoreOS (RHCOS).

The scope of this document is limited to OpenShift Virtualization as an OpenShift extension installed on top of the platform. Additionally, guest operating systems deployed using OpenShift Virtualization must be separately validated by adhering to the hardening procedures that are outlined in each operating systems' respective specifications.

Although this document follows the formatting conventions of the Center for Internet Security (CIS), it is not affiliated with CIS. Red Hat is in the process of formalizing these recommendations as an OpenShift Virtualization CIS benchmark.

Red Hat

Contents

Red Hat

Platform Configuration

OpenShift platform configuration is important for security because it enables the administrator to define and enforce the policies and procedures that will guide how applications are built, deployed, and managed on the platform.

Restrict GPU and USB pass through to approved devices

Profile Applicability: Level 1

Description: The ability to pass through devices provides the capability to offload tasks from the CPU to the device itself.

Rationale: Restricting passthrough to approved devices reduces the risk of unauthorized or unintended data sharing/transmission introduced by allowing GPU and USB connectivity.

Impact: Unauthorized devices will not be passed through, and therefore not be available to workloads.

Audit: List the host devices available to virtualization workloads and verify the host devices returned are necessary for workload execution.

```
$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv
-ojsonpath='{.spec.permittedHostDevices}'
```

Remediation: Enforcing limitations to the passthrough of approved devices protects against the introduction of a malicious or unknown device to the system.
The system administrator can remove the unwanted devices by patching the HCO object.

Example: Assuming the following output from the audit command above:

```
            {
"mediatedDevices": [
  {
    "mdevNameSelector": "DEVICE_A",
    "resourceName": "vendor.com/DEVICE_A"
  }
],
"pciHostDevices": [
  {
    "externalResourceProvider": true,
```

```
      "pciDeviceSelector": "XXX:YYY",
      "resourceName": "vendor.com/DEVICE_B"
    },
    {
      "pciDeviceSelector": "NNN:MMM",
      "resourceName": "vendor.com/DEVICE_C"
    }
  ],
  "usbHostDevices": [
    {
      "resourceName": "kubevirt.io/storage",
      "selectors": [
        {
          "product": "0001",
          "vendor": "46f4"
        }
      ]
    }
  ]
```

If the first PCI device (the DEVICE_A at index 0, in bold) should not be permitted to be passed through to a VM, here's how to remove it:

```
$ oc patch hyperconverged kubevirt-hyperconverged -n openshift-cnv
--type='json' -p='[
    {"op": "remove", "path":
"/spec/permittedHostDevices/pciHostDevices/0"},
         ]'
```

To remove all permitted devices, use the following:

```
$ oc patch hyperconverged kubevirt-hyperconverged -n openshift-cnv
--type='json' -p='[
    {"op": "remove", "path": "/spec/permittedHostDevices"},
]'
```

Default Value: By default, OpenShift Virtualization doesn't configure any pass through devices.The result of the command should be `empty`.

Enable nonRoot feature gate in OCPvirt prior to 4.18

Profile Applicability: Level 1

Description: Root is a default account with elevated privileges, including full administrative access to both security and non-security related functions. The "nonRoot" feature gate in OpenShift environments ensures virtual machine access is restricted to resources and functionality required to operate.

Rationale: Unauthorized access to a root account without restrictions implemented by the nonRoot feature introduces the risk of unintended or unauthorized access to privilege elevation and the ability to perform administrative tasks. This feature is a foundational mechanism in implementing role and attribute based access controls.

Impact: Disabling root introduces the need to implement alternate identification and authentication methods to access administrative functions.

Audit: Use the following command to list `nonRoot` settings for hyper converged resources:

```
$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv
-ojsonpath='{.spec.featureGates.nonRoot}'
```

Verify the output is `true`. nonRoot is deprecated in Openshift Virtualization from 4.18, and the secure default is enforced.

Remediation: Enable `nonRoot` and set its value to `True`.

```
$ oc patch hyperconverged kubevirt-hyperconverged -n openshift-cnv
--type='json' -p='[
 {"op": "replace", "path": "/spec/featureGates/nonRoot", "value":
true },
]'
```

Default Value: By default, KubeVirt workloads run in nonRoot mode and it sets to false. From 4.18, this field will be ignored, and always be enabled.

Disable persistentReservations feature gate

Profile Applicability: Level 1

Description: Persistent reservations are used to reserve a shared LUN among multiple virtual machines. This feature is required when the Windows guest makes use of the Windows Shared Cluster Filesystem. Enabling persistent reservations introduces additional complexity and overhead, as it requires calculated management of resource allocation and monitoring to ensure that reservations are not overcommitted and do not impact the overall performance of the system.

Enabling the persistent reservations feature introduces and relies upon an optional component with elevated privileges (qemu-pr-helper).

It is recommended to disable persistent reservations if you are not planning to use Windows Shared Cluster Filesystem for any virtual machines running on your OpenShift Virtualization environment.

Rationale: Persistent reservations in Virtualization are only required in specific use-cases. Restricting enabled features aligns with the concept of least functionality, reducing the operating risk a component introduces to a system.

Impact: Disabling this feature will prevent the operation of workloads that rely on Windows Shared Cluster Filesystem.

Audit: Use the following command to list persistent reservation settings for hyperconverged resources:

```
$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv
-ojsonpath='{.spec.featureGates.persistentReservation}'
```

It should return false.

Remediation: Disable the persistent reservations feature by setting `persistentReservations` to `False` for applicable hyperconverged resources.

```
$ oc patch hyperconverged kubevirt-hyperconverged -n openshift-cnv
--type='json' -p='[
 {"op": "replace", "path":
"/spec/featureGates/persistentReservation", "value": false },
]'
```

Default Value: By default `persistentReservations` is disabled and set to `false`. It is recommended to be enabled only when Windows Shared Cluster Filesystem or similar applications are planned to be used.

Disable downwardMetrics feature gate

Profile Applicability: Level 1

Description: The `downwardMetrics` feature allows users to collect and monitor additional metrics related to host and virtual machine performance.

Rationale: Enabling the downwardMetrics feature introduces the risk of unauthorized or unintended sharing of information, as well as manipulation of information flow restrictions. The additional metrics include sensitive performance data, which could aid in the reconnaissance activities of a malicious actor.

Impact: Guests are not able to use the host metrics.

Audit: Check the the downwardMetrics feature gate is disabled

```
$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv
-ojsonpath='{.spec.featureGates.downwardMetrics}'
```

Verify the output is `false`

Remediation: Set `downwardMetrics` to `False`.

```
$ oc patch hyperconverged kubevirt-hyperconverged -n openshift-cnv
--type='json' -p='[
 {"op": "replace", "path": "/spec/featureGates/downwardMetrics",
"value": false },
]'
```

Default Value: By default, the `downwardMetrics` is set to `false`.

Enforce the use of trusted registries using TLS

Profile Applicability: Level 1

Description: Transport Layer Security (TLS) is a cryptographic protocol used in this context to protect data in transit. Restricting operations to the use of trusted registries ensures the use of approved container images.

Rationale: By only pulling container images from trusted registries, organizations can reduce the risk of introducing unknown vulnerabilities or malicious software into their systems. This helps ensure that their applications and systems remain secure and stable.

Impact: Enforcing trusted container image registries limits which container images users are able to deploy and manage. This restriction may also limit the use of some platforms and/or tools.

Audit: Use the following command to view any insecure registries:

```
$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv
-ojsonpath='{.spec.storageImport.insecureRegistries}'
```

Verify no registries are returned.

Remediation: Do not allow connections to insecure registries from hyperconverged resources. The cluster administrator can remove any insecure registry:

```
$ oc patch hyperconverged kubevirt-hyperconverged -n openshift-cnv
--type='json' -p='[
 {"op": "remove", "path": "/spec/storageImport/insecureRegistries"},
]'
```

Default Value: TLS is enabled by default. No insecure registries are configured or enabled by default. The result of the command should be empty.

Restrict patching operations in the annotations for Hyperconverged

Profile Applicability: Level 1

Description: The kubevirt-hyperconverged (HCO) object provides a mechanism to customize OpenShift Virtualization components through the control of patching operations. Although this approach itself does not inherently introduce security risks, enabling experimental features may have unintended consequences and might not be officially supported. It is essential to weigh the benefits of using these options against any potential security implications before proceeding.

Rationale: Cloud administrators should not patch OpenShift Virtualization objects, but rather configure the OpenShift Virtualization options available in HCO.

Impact: Restrictions to patching operations may prevent the use of experimental features not yet supported or considered insecure.

Audit: Ensure there are no patching operations for the Openshift Virtualization component

```
$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv -o
jsonpath='{.metadata.annotations}'| jq
'.|has("kubevirt.kubevirt.io/jsonpatch")'

$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv -o
jsonpath='{.metadata.annotations}'| jq
'.|has("containerizeddataimporter.kubevirt.io/jsonpatch")'

$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv -o
jsonpath='{.metadata.annotations}'| jq
'.|has("networkaddonsconfigs.kubevirt.io/jsonpatch")'

$ oc get hyperconverged kubevirt-hyperconverged -n openshift-cnv -o
jsonpath='{.metadata.annotations}'| jq
'.|has("ssp.kubevirt.io/jsonpatch")'
```

All the commands should return an empty string.

Remediation: The annotations should be removed from the Hyperconverged. For example, by directly editing the object with `oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv` or by removing the annotation with the `annotate` command.

Example:

```
$ oc annotate --overwrite -n openshift-cnv hco
kubevirt-hyperconverged
'containerizeddataimporter.kubevirt.io/jsonpatch-'
```

Default values: None of the annotations should be set and the result of the commands should always be `false`.

Ensure KSM is disabled

Profile Applicability: Level 2

Description: Kernel Samepage Merging (KSM) is a Linux kernel feature that merges identical memory pages across virtual machines running on a single host, potentially reducing memory usage and improving system performance.

Rationale: Sharing resources introduces the risk of memory corruption attacks, unauthorized or unintended sharing of information, and the manipulation of data flow restrictions.

Impact: Disabling Kernel Samepage Merging (KSM) does not impact normal node operations. Workloads requiring a high amount of memory usage could potentially see an impact on system performance.

Audit: Use the following command to list KSM configuration for hyperconverged resources:

```
$ oc get hyperconvergeds kubevirt-hyperconverged -n openshift-cnv
-ojsonpath='{.spec.ksmConfiguration}'
```

Remediation: Disable the KSM feature.For example, by directly editing the object with `oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv` and set the field to `false`.

Default: KSM in disabled by default. The command should return false.

Ensure kubevirt seccomp profile file permission is set to 700 or more restrictive

Profile Applicability: Level 1

Description: Setting the `kubevirt` Secure Computing Mode (seccomp) file permissions to 700 or more restrictive limits access to security related administrative functionality to approved administrators.

Rationale: This configuration ensures that only authorized users have access to modify the seccomp profile, thereby preventing unauthorized alterations to the allowed system calls for virt-launcher pods. By enforcing stringent permission controls, potential security risks associated with unrestricted access to sensitive system calls can be mitigated.

Impact: None.

Audit: Verify seccomp file permissions for the `virt-launcher` pod, run the following command:

```
$ for pod in $(oc get pod -n openshift-cnv -l
kubevirt.io=virt-handler --no-headers -o
custom-columns=":metadata.name"); do
  oc exec -ti -n openshift-cnv $pod -c virt-handler -- stat -c %a
/proc/1/root/var/lib/kubelet/seccomp/kubevirt/kubevirt.json
done
```

Should return 700 or if the file doesn't exist then the corresponding feature gate is not enabled and this check can be safely ignored.

Remediation: Update the permissions on each node using `chmod`:

```
$ oc exec <virt-handler-pod> -ti  -n openshift-cnv -- chmod 700
/proc/1/root/var/lib/kubelet/seccomp/kubevirt/kubevirt.json
```

Default value: If the Secure Computing Mode feature is enabled, the permissions of this file are 700 by default.

Ensure kubevirt cache directory permission is set to 755 or more restrictive

Profile Applicability: Level 1

Description: Setting the `kubevirt` private directory to 755 or more restrictive limits access to security related administrative functionality to approved administrators.

Rationale: This configuration ensures that the directory where virt-handler stores the virtual machine configuration for caching is restricted to administrators and Openshift Virtualization privileged node components.

Impact: None.

Audit: Directory permissions for the `virt-handler` pods, run the following command:

```
$ for pod in $(oc get po -n openshift-cnv -l kubevirt.io=virt-handler
--no-headers  -o custom-columns=":metadata.name"); do   oc exec $pod
-ti  -n openshift-cnv -- stat -c %a /var/run/kubevirt-private ;done
```

Should return 755 or if the file doesn't exist then the corresponding feature gate is not enabled and this check can be safely ignored.

Remediation: Update the permissions on each virt-handller using chmod:

```
$ oc exec <virt-handler-pod> -ti  -n openshift-cnv -- chmod 755
/var/run/kubevirt-private
```

Default value: The default permissions of this directory are 755.

Restrict namespace administrator access to migration tools

Profile Applicability: Level 1

Description: Namespace administrators have the capability to perform live migrations, which can potentially impact other users of the system. Live migration is a process where virtual machines can be moved from one physical host to another without any downtime, making it a valuable tool for managing clusters and ensuring high availability. However, since live migration involves moving data between hosts, it can have an impact on other users who may have applications running on the same hosts. As such, it is important to exercise caution when granting access to namespace administrators and ensure that appropriate safeguards are in place to minimize any potential disruptions.

Rationale: Limiting the number of virtual machine migrations occurring simultaneously can help to ensure that the system remains stable and performs optimally during periods of high migration activity. If too many migrations are occurring at once, it can lead to increased resource usage, reduced processing power, and potential network congestion, all of which can impact system performance and availability.

The KubeVirt community is working towards removing migration functionality from namespace administrators in order to make it safer and more manageable. Live migration, on the other hand, will be reserved for cluster administrators. This separation of duties helps ensure that live migrations can be performed safely and without impacting other users of the system.

Please refer to the upstream proposal for more information.

Impact: Using `VirtualMachineInterfaceMigrations` can introduce additional overhead to the system, particularly if a large number of migrations are occurring simultaneously. This can lead to increased resource usage and potential performance degradation, as the migration process requires significant processing power and network bandwidth. Additionally, if a migration fails or encounters errors during the migration process, it can cause downtime for the associated VM and potentially impact system availability.

Audit: Use the following command to find users, service accounts and groups who are allowed to create `VirtualMachineInterfaceMigration` and `MigrationPolicy` resources:

```
$ oc adm policy who-can create vmim
```

```
$ oc adm policy who-can create migrationpolicy
```

Ensure users are authorized to perform those actions.

Remediation: Remove `create` access to virtualmachineinstancemigration and migrationpolicy objects in the cluster.

Example: Remove `create` access given by the `clusterRoleBinding` for the I `migrationpolicy` for the `test` user

```
# Get all the users and service accounts who can create
migrationpolicies
$ oc adm policy who-can create migrationpolicy
Users:  system:admin
..
        Test
# Verify that the test user can create the migrationpolicy
$ oc auth can-i create migrationpolicies --as test
Warning: resource 'migrationpolicies' is not namespace scoped in group
'migrations.kubevirt.io'
yes

# Find out which rolebinding or clusterrolbinding associated to the
test user
$ oc get rolebindings,clusterrolebindings   --all-namespaces    -o
custom-columns='KIND:kind,NAMESPACE:metadata.namespace,NAME:metadata.n
ame,SERVICE_ACCOUNTS:subjects[?(@.kind=="User")].name' |grep test
ClusterRoleBinding   <none>
migration-creator
test

# Inspect the cluster role binding
$ oc get clusterrolebindings migration-creator -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
```

```
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"rbac.authorization.k8s.io/v1","kind":"ClusterRoleBindin
g","metadata":{"annotations":{},"name":"migration-creator"},"roleRef":
{"apiGroup":"rbac.authorization.k8s.io","kind":"ClusterRole","name":"m
igration-creator"},"subjects":[{"apiGroup":"rbac.authorization.k8s.io"
,"kind":"User","name":"test"},{"kind":"ServiceAccount","name":"test","
namespace":"default"}]}
  creationTimestamp: "2025-03-06T14:05:04Z"
  name: migration-creator
  resourceVersion: "1093678"
  uid: 96be5dc2-2b30-4734-b5ef-16d9342bbdbf
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: migration-creator
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: test
- kind: ServiceAccount
  name: test
  namespace: default

# Remove the cluster role binding
$ oc delete clusterrolebindings migration-creator
clusterrolebinding.rbac.authorization.k8s.io "migration-creator"
deleted

# Re-verify that the test user cannot create the migrationpolicy
$ oc auth can-i create migrationpolicies --as test
Warning: resource 'migrationpolicies' is not namespace scoped in group
'migrations.kubevirt.io'

no
```

Default Value: By default, cluster administrators and namespace administrators can trigger virtual machine migrations.

Restrict exec access to the pods

Profile Applicability: Level 1

Description: The ability to `exec` commands in a pod allows for arbitrary execution by users. This includes administrative functions which normally require elevation of privileges by an approved administrator, and could lead to unauthorized use of both security and non-security related administrative functions.

Rationale: The exec command is not necessary for the proper functioning of OpenShift Virtualization.

Impact: Limiting access to exec can restrict access to utilities used to accomplish tasks users are authorized to perform. These restrictions may require more granular role or attribute based access controls to be defined.

Audit: To verify who can exec commands in pods, use the following command:

```
$ oc adm policy who-can exec pod
```

Remediation: Assign `exec` access to `pods` in the cluster only to approved administrators.

Default Value: The ability to run `exec` commands is reserved for cluster administrators by default.

Restrict VNC access to cluster workloads

Profile Applicability: Level 1

Description: Access to each workload's virtual screen is provided via VNC. Permission to use VNC is granted via a role, which means users with role assigned can access the VNC console of all workloads in a namespace.

Rationale: Access to use VNC should be carefully considered as it is usually preferable to access workloads via other means which support a stronger encryption and authentication mechanism.

Impact: Limiting VNC access to cluster workloads means users will need to access the machine using approved communication mechanisms.

Audit: Verify to which users have the necessary permissions to create tokens:

```
$ oc get rolebinding -ojson  | jq -c '.items[] | select(.roleRef.name
| contains("token.kubevirt.io:generate"))'
```

Remediation: Assign this RoleBinding only to users with approval to use it, otherwise remove unapproved users.

Default: No default value, the command returns the list of rolebindings which has the roleName to generate the kubevirt token. If none was created then the command returns an `empty` list.

Restrict access to create and modify the Virtual Machine Cluster Instance and Preference Types

Profile Applicability: Level 1

Description: An instance type is a reusable object used to create VMs with sane defaults based on the desired type of VM. Since the object is used to create instances of VMs in the entire cluster, its creation and update needs to be carefully and securely managed.

Rationale: Because an instance type is used as a starting point to create multiple copies of other VMs, a mistake or insecure setting in the instance type will be propagated to all new VMs based on it.

Impact: Workload owners who require the ability to create and modify functionality will be granted permission on an as-needed basis.

Audit: Check who has the ability to create InstanceTypes with the following command:

```
$ oc adm policy who-can create,update
virtualmachineclusterinstancetypes
```

Remediation: Assign `create` and `update` access only to approved `virtualmachineclusterinstancetypes` objects in the cluster. Remove the create and update permission from users who are not authorized to perform the actions.

Default Value: This permission is granted only to cluster admins by default.

Restrict update access to the CDI CR

Profile Applicability: Level 1

Description: The `CDI` object is not namespaced and contains sensitive configuration data for the Containerized Data Importer.

Rational: Reduce at the minimum who can modify `CDI` resources.

Impact: Workload owners don't normally need this feature.

Audit: check who can update the `cdi` resources:

```
$ oc adm policy who-can update cdi
```

Remediation: Assign `update` access only to approved `cdi` objects in the cluster.

Default Value: This permission is not granted by default.

Virtual Machine Configuration

Optimized hypervisor or virtualization configuration enables organizations to create a virtualized environment that segregates workloads and applications from one another. Proper configuration can reduce the risk of attacks that could compromise multiple systems at once, as well as prevent unauthorized access to sensitive data by restricting communication between system components. Additionally, OpenShift Virtualization includes security features at the platform level, such as encryption data at rest, data in transit, and access controls that can be used to further protect against threats.

Only when the workload is generated by the users—which could occur at any time—can virtual machine validations be performed. Furthermore, a virtual machine may be in a running or stopped state, and as the information is accessible only when the guest is launched, some tests may only be carried out while the virtual machine is operating.

Therefore the virtual machine validation needs to either be launched periodically to include newly created guests or it needs to be triggered by a virtual machine creation event.

Restrict pass through of GPUs and Host devices to the Virtual Machine

Profile Applicability: Level 1

Description: The ability to pass through certain devices like a GPU or NVMe can improve performance by offloading certain tasks from the CPU to the device itself, but it also introduces shared system resources and additional attack vectors for bad actors. Configuring GPUs or host devices for the virtual machine can put the guest at risk. It's important to be selective and only enable passthrough for devices that are critical to the operation of your workloads.

Rationale: GPUs and host devices are directly passed to the Virtual Machine for better performance, but it also constitutes a risk if the device has been infected

Impact: Avoiding the usage of GPUs and host devices can prevent the users to run high performance workloads or achieving good performance for NVMe devices

Audit: Use the following command to list `gpus` associated with a virtual machine:

```
$ oc get vm -A -ojson | jq '.items[] |
select(.spec.template.spec.domain.devices.gpus | length > 0) |
{vm_name: .metadata.name, namespace: .metadata.namespace, gpus:
.spec.template.spec.domain.devices.gpus}'
```

Use the following command to list `hostDevices` associated with a virtual machine:

```
$ oc get vm -A -ojson | jq
'.items[].spec.template.spec.domain.devices.hostDevices'
```

This check applies to `any created` VMs.

Results of this query will be in this form:

```
[
  {
    "deviceName": "<resource-name>",
    "name": "<device-name>"
  },
  {
    "deviceName": "<resource-name>",
    "name": "<device-name>"
  }
]
```

Remediation: Only enable approved devices which are trust-worthy and strictly necessary by the workload.
For example, if the first host device above (index 0, in bold) should not be permitted to be passed through to a given VM, here's how to remove it:

```
$ oc patch vm <vm-name> --type='json' -p='[
    {"op": "remove", "path":
"/spec/template/spec/domain/devices/hostDevices/0"},
]'
```

Default Value: By default, no GPUs or hostDevices are enabled by default on the VM.spec. The command should return an `empty` list.

Disable overcommitting guest memory

Profile Applicability: Level 1

Description: The `overcommitGuestOverhead` configuration option enables the request for additional virtual machine management memory inside the `virt-launcher` pod. The overcommit feature is used to increase virtual machine density on the node, as long as the virtual machine doesn't request all the memory that it would need if fully loaded. However, if the VM were to use all of the memory it could, this would lead to the OpenShift Scheduler killing the workload.

Rationale: This is a hypervisor-level feature that allows nodes to host more virtual machines than would normally be allowed by KubeVirt's scheduling algorithms. If the VM consumes the entire memory might cause the guest to crash with workload interruptions and guest malfunctioning.

Impact: `overCommitGuestOverhead` is used to (unsafely) increase workload density, meaning more virtual machines might be able to run on a node than otherwise.

Audit: Verify that no virtual machines are using the `overcommitGuestOverhead` setting:

```
$ oc get vm -ojson -A |jq '.items[] |
select(.spec.template.spec.domain.resources.overcommitGuestOverhead ==
true)| .metadata.namespace + "/" + .metadata.name'
```

Remediation: Stop any guest with this flag enabled and remove the flag before restarting. This flag can be removed on a given VM with the following command:

```
$ oc patch vm <vm-name> --type='json' -p='[
    {"op": "remove", "path":
"/spec/template/spec/domain/resources/overcommitGuestOverhead"},
]'
```

Default Value: OpenShift Virtualization disables `overcommitGuestOverhead` by default and the list of virtual machines returned by the command should be `empty`.

Storage Components

Storage configuration is important for security in virtualized environments because it enables organizations to define and enforce policies that control how data is stored and accessed. Proper configuration can help prevent unauthorized access to sensitive data by limiting who has access to what information and ensuring that data is stored in a secure manner. Additionally, storage platforms often include security features such as encryption and access controls that can be used to further protect against threats.

Restrict access to cross datavolumes cloning

Profile Applicability: Level 1

Description: Datavolume cross-namespace cloning allows cloning a `datavolume` from one namespace to another, breaking the namespace isolation.

Rationale: Any user with access to multiple namespaces for the purpose of cloning a `datavolume` consequently has access to the underlying data of a volume they do not own.

Impact: Limiting access to namespaces means cross namespace datavolume cloning will not be possible.

Audit: To check which role bindings have the ability to clone across namespaces, use the following command where to list all the rolebinding and verify which one has a ClusterRole which enables it to operate on data volumes and have bound a service account in another namespace. Ensure that the destination namespace is a desired one.

```
oc get rolebinding -n <source-namespace> <allow-clone-to-user> -oyaml
```

Output will be in the following format:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: <role-binding-name>
  namespace: <source namespace>
```

```
subjects:
- kind: ServiceAccount
  name: default
  namespace: <destination namespace>
roleRef:
  kind: ClusterRole
  name: <datavolume-cloner>
  apiGroup: rbac.authorization.k8s.io
```

Example:

```
$ oc get clusterRole datavolume-cloner -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"rbac.authorization.k8s.io/v1","kind":"ClusterRole","meta
data":{"annotations":{},"name":"datavolume-cloner"},"rules":[{"apiGroup
s":["cdi.kubevirt.io"],"resources":["datavolumes/source"],"verbs":["*"]
}]}
  creationTimestamp: "2025-02-06T14:51:36Z"
  name: datavolume-cloner
  resourceVersion: "9267"
  uid: 4ad67fec-0461-4cbe-b460-a0fb5533be6d
rules:
- apiGroups:
  - cdi.kubevirt.io
  resources:
  - datavolumes/source
  verbs:
  - '*'
$ oc get rolebinding -n src-ns -ojson
apiVersion: v1
items:
- apiVersion: rbac.authorization.k8s.io/v1
```

```
  kind: RoleBinding
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"rbac.authorization.k8s.io/v1","kind":"RoleBinding","meta
data":{"annotations":{},"name":"datavolume-cloner","namespace":"src-ns"
},"roleRef":{"apiGroup":"rbac.authorization.k8s.io","kind":"ClusterRole
","name":"datavolume-cloner"},"subjects":[{"kind":"ServiceAccount","nam
e":"default","namespace":"ds-ns"}]}
    creationTimestamp: "2025-02-06T14:53:07Z"
    name: datavolume-cloner
    namespace: src-ns
    resourceVersion: "9583"
    uid: 7f633396-27b3-463d-83a8-103c6d3d5888
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: datavolume-cloner
  subjects:
  - kind: ServiceAccount
    name: default
    namespace: ds-ns
kind: List
metadata:
  resourceVersion: ""
```

The clusterRole `datavolume-cloner` allows copying the datavolume source. In the example, the command lists all the rolebindings in the source namespace and one of them enables the `default` service account in the namespace `dst-ns` to clone all the data volumes from the namespace `src-ns`.

Remediation: Remove any `rolebinding` resources that grant unintended access across namespaces.

Please refer to the latest OpenShift documentation for details on this process.

Default Value: Data volume cloning is limited to cluster administrators by default. Delegating the migration of data volumes across namespaces requires a cluster administrator to create new roles and role bindings for users who require that functionality for their job responsibilities.

Disable Shareable disks

Profile Applicability: Level 1

Description: Sharable disks are shared between VMs, and they can be used only if a filesystem installed on top of it is a cluster-file system or the application using the device is distributed and cloud aware. Their wrong usage might cause data corruption.

Rationale: Sharing system resources increases the risk of unauthorized access to data, manipulation of data flow restrictions, and could lead to corruption and data loss.

Impact: Sharable disks are one means of sharing data between workloads, which cannot be used if this feature is avoided.

Audit: Inspect virtual machine volumes and ensure the shareable flag is not set to `true`:

```
$ oc get vm -ojson -A | jq '.items[] |
select(.spec.template.spec.domain.devices.disks[].shareable == true)|
.metadata.namespace + "/" + .metadata.name'
```

Remediation: Remove the shareable flag from any disk where it is found.
For example, to remove the shareable flag on the first disk (index 0) of a given VM, this command can be used:

```
$ oc patch vm <vm-name> --type='json' -p='[
    {"op": "remove", "path":
"/spec/template/spec/domain/devices/disks/0/shareable"},
]'
```

To set a shareable flag to false on every disk, this (experimental) command can be used:

```
$ oc get vm <vm-name> -o json | jq
'.spec.template.spec.domain.devices.disks[] += {"shareable":false}' |
oc apply -f -
```

Default Value: Shareable disks are not enabled by default. The command should return an empty list.

Ensure errorPolicy is not set to ignore

Profile Applicability: Level 1

Description: Error policies for disks control how Input/Output (IO) errors are handled. If a read or write operation fails on the storage, an IO error occurs. Setting the error policy to `ignore` is not recommended because it makes it difficult to determine the root cause of any IO issues, which can lead to data loss if the data is not correctly written to the disk.

Rationale: Certain applications, such as Windows Shared Cluster Filesystem, require a different behavior than the default error policy setting. IO errors need to be reported to the guest Operating System by setting the `errorPolicy` to `report`.

Impact: Emergency or maintenance activities may require reducing log generation by temporarily ignoring the IO errors. This may be accomplished by setting the `errorPolicy` to `ignore`.

Audit: Verify that no virtual machines are using the `errorPolicy` set to `ignore`:

```
$ oc get vm -ojson -A | jq '.items[] |
select(.spec.template.spec.domain.devices.disks[].errorPolicy ==
"ignore")| .metadata.namespace + "/" + .metadata.name'
```

Remediation: Remove the `errorPolicy` or set it to `stop` (default value) or `report`(suggested option for Windows Shared Cluster Filesystem).
For example, to remove the errorPolicy flag on the first disk (index 0) of a given VM, this command can be used:

```
$ oc patch vm <vm-name> --type='json' -p='[
    {"op": "remove", "path":
"/spec/template/spec/domain/devices/disks/0/errorPolicy"},
]'
```

To set an errorPolicy flag to stop on every disk, this (experimental) command can be used:

```
$ oc get vm <vm-name> -o json | jq
'.spec.template.spec.domain.devices.disks[] += {"errorPolicy":"stop"}'
| oc apply -f -
```

Default Value: By default the `errorPolicy` is not set on any disk and the default value is `stop`.
The audit procedure should return an empty list.

Networking Components

Networking configuration is important for security in virtualized environments because it enables organizations to define and enforce policies that control how data is transmitted between systems. Proper configuration can help prevent unauthorized access to sensitive data by segregating networks and segmenting them into smaller, more contained areas. Additionally, networking platforms often include security features such as encryption and access controls that can be used to further protect against threats.

Use dedicated VLANs to segment network traffic

Profile Applicability: Level 2

Description: To ensure that virtual machines cannot access each other's network interfaces, it is crucial to segment them completely using separate VLANs. By connecting virtual machines to logically separated VLANs, you enforce network segmentation at the L2 level, preventing virtual machines from unauthorized/unintended data transmission, and reducing the risk of security breaches or data leaks.

Rationale: The importance of layer 2 network segmentation can be attributed to its ability to enhance network security by creating separate broadcast domains for different groups of devices on the network. These broadcast domains limit the spread of broadcast traffic, making it more difficult for malicious actors to propagate their attacks across the entire network.

Impact: Segmenting workloads to dedicated VLANs also means they can't access services provided by other workloads unless they are in the same VLAN.

Audit: Use the following command to determine if VLANs are configured for OVN Kubernetes:

```
$ oc get networkattachmentdefinition -o json | jq
".items[].spec.config"
```

Review the output to determine if VLANs are configured correctly:

```
config: |-
  {
    "type": "ovn-k8s-cni-overlay",
```

```
      "topology": "localnet",
      "vlanID": 100
  }
```

Please refer to the latest OpenShift documentation for more information.

Use the following command to determine if VLANs are configured for SR-IOV:

```
$ oc get sriovnetwork -o json | jq ".items[].spec.vlan"
```

Output will return `null` for networks without VLANs configured, or a VLAN ID (e.g., `100`).

Please refer to the latest OpenShift documentation for more information.

Use the following command to determine if VLANs are configured for Bridge CNI:

```
$ oc get networkattachmentdefinition -o json | jq
".items[].spec.config"
```

Review the output to determine if VLANs are configured correctly:

```
config: |
  {
    "type": "bridge",
    "vlan": 100,
    "preserveDefaultVlan": false
  }
```

Please refer to the latest OpenShift documentation for more information.

This is an opt-in feature. No virtual machines will have a dedicated VLAN unless configured per the networking provider instructions.

Remediation: Use dedicated VLANs as required. Please refer to the latest OpenShift documentation for the appropriate networking provider. Documentation: SR-IOV CNI, VLAN with bridge CNI, VLAN with OVN localnet.

Default Value: Dedicated VLANs are not used by default.

Enable MAC spoof filtering

Profile Applicability: Level 1

Description: Mac spoof filtering is a feature that helps prevent man-in-the-middle attacks by verifying the authenticity of MAC addresses in network packets. By checking the MAC address of incoming packets, mac-spoof filtering can help ensure that only authorized devices are able to communicate with a particular device or network.

Rationale: MAC spoof filtering prevents unauthorized access to system components and data.

Impact: None.

Audit: Enable MAC spoof filtering, allowing a given virtual machine to only connect to the network from its allocated MAC address.

For SR-IOV, use the following command:

```
$ oc get sriovnetwork -o json | jq ".items[].spec.spoofChk"
```

Verify the output is on for each network. Please refer to the latest OpenShift documentation for more information.

For Bridge CNI, use the following command:

```
$ oc get networkattachmentdefinition -o json | jq
".items[].spec.config.macspoofchk"
```

Verify the output is `true` for each network attachment definition. Please refer to the latest OpenShift documentation for more information.

Remediation: Enable MAC-spoof filtering per the deployed networking provider instructions. Instructions for Bridge CNI and SR-IOV CNI
Default Value: OVN Kubernetes localnet enables MAC spoof filtering by default.

Use multi-network policies

Profile Applicability: Level 2

Description: Micro-segmentation in VLANs is crucial for enhancing network security because it allows control of traffic flow between different components of the network and their services. This segmentation helps to contain potential security breaches and limit the spread of malware or unauthorized access to information.

Rationale: Micro-segmentation enables better control over access privileges and traffic flow, allowing administrators to implement restrictive security policies and identify anomalous network activity more effectively.

Impact: Once there is any Multi-network Policy in a namespace selecting a particular VM and the network it's connected to, that VM will reject any connections from the given network that are not allowed by any Multi-network Policy.

Audit: By default, no MultiNetworkPolicies are defined. To segregate one or more VMs/Pods in a project, you create MultiNetworkPolicy objects in that project to indicate the allowed incoming connections on a given network. Project administrators can create and delete MultiNetworkPolicy objects within their own project.

Run the following command and review the MultiNetworkPolicy objects created in the cluster.

```
$ oc -n all get multi-networkpolicy
```

Ensure that each namespace defined in the cluster has at least one `Multi-network Policy`.

Remediation: Follow the documentation linked below and create `Multi-NetworkPolicy` objects as you need them.

Note: A Multi-network Policy must match a VM and also the `NetworkAttachmentDefinition` this VM is connected to.

Default Value:
Use MultiNetworkPolicies to configure micro segmentation within a VLAN. This feature is only available with OVN Kubernetes localnet. This feature is disabled by default.

See OpenShift docs

Host Firmware (any options which need to be disabled/enabled for KVM)

Physical host firmware configuration is important for security in virtualized environments because it enables organizations to define and enforce policies that control how hardware resources are managed and accessed. Proper configuration can help prevent unauthorized access to sensitive data by ensuring that only authorized users have access to physical hosts, and that only necessary services are running on those hosts. Additionally, physical host firmware platforms often include security features such as encryption and access controls that can be used to further protect against threats.

The host firmware is a component under the control of the cluster-administrator. Thus, the following checks and suggestions are on top of the hardening techniques that must be implemented to the system configuration document by the hardware vendor.

Disable Intel Trusted Execution Technology (TXT)

Profile Applicability: Level 1

Description:
Intel Trusted Execution Technology (TXT) is a security-related CPU feature that is not currently leveraged by OpenShift or Openshift virtualization.

Rationale: Neither OCP nor Openshift Virtualization leverage Trusted Execution Technology, which can be safely disabled.

Impact: None.

Audit: Use the following command to return the BIOS settings for a physical host:

```
$ fwupdmgr get-bios-settings
```

Look for `TxT`, and ensure the `Current value` is `Disabled`.

Remediation:
Disable TXT via the firmware settings of each worker node.

Default Value:

The default value is firmware-dependent.

Host Kernel

Kernel configuration is important for security in virtualized environments because it enables organizations to define and enforce policies that control how the operating system interacts with hardware resources. Proper configuration can help prevent unauthorized access to sensitive data by ensuring that only necessary services are running on hosts, and that only authorized users have access to those services. Additionally, kernel platforms often include security features such as encryption and access controls that can be used to further protect against threats.

The host kernel is more a component under the controller of RHCOS and OCP installation rather than Openshift Virtualization. Therefore, the following checks and suggestions should be implemented in addition to the hardening techniques for the host operating system.

Disable nested virtualization

Profile Applicability: Level 1

Description:
Enabling nested virtualization in OpenShift Virtualization allows virtual machines to run within a hypervisor that is itself running on top of an existing operating system. This setup, known as nested virtualization, enables the VMs to access the full range of virtualization features provided by the underlying host, including access to the virtualization stack. In this case, Openshift Virtualization uses nested virtualization to provide its VMs with access to the entire virtualization stack, allowing them to take advantage of advanced virtualization capabilities such as live migration, resource pooling, and cloning. Nest virtualization is required if the cluster admin expects users to run Windows guest and Windows Subsystem for Linux (WSL) 2.

Rationale:
While nested virtualization has safeguards in place to prevent malicious activity, disabling it undeniably reduces the attack surface.

Impact:
Nested virtualization is required for special use-cases when cluster-admins expect to use virtualization inside the guests i.e. if the VMs in the cluster may need virtualization, for WSL2 or other purposes.

Note: If nested virtualization is disabled, Openshift Virtualization VMs will not be able to use hardware virtualization to start their own VMs. Windows guests will also be unable to use WSL2.

Audit: To check the state of nested virtualization on nodes running on Intel CPUs:

```
$ cat /sys/module/kvm_intel/parameters/nested
```

Disabled is denoted by `0`, and enable is denoted with `1`.

If you're using AMD CPUs, replace `kvm_intel` with `kvm_amd`.

Remediation: Add the appropriate kernel arguments to the MachineConfigPools of the workers nodes and reboot them. See https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html/installation_configuration/installing-customizing#installation-special-config-kargs_installing-customizing

Default Value: OpenShift does not enable nested virtualization by default.

Enable vCPU metrics

Profile Applicability: Level 1

Description: Openshift Virtualization provides metrics that you can use to monitor the consumption of cluster infrastructure resources, including vCPU. In order to use the vCPU metric, the `schedstats=enable` kernel argument must be applied.

Rationale: Metrics are a fundamental mechanism for understanding if the guest is behaving in an anomolous suspicious manner.

Impact: The kernel option is a prerequisite for using the prometheus metrics for the vCPU usage, without the metrics being reported.

Audit: Verify if the option is enabled in the kernel command line:

```
$ cat /proc/cmdline | grep schedstats=enable
```

Remediation: Enable the kernel option on all worker nodes as documented in Chapter 1. Customizing nodes | Red Hat Product Documentation

Default Value: OpenShift enables `schedstats` by default. The command should returns a string containing `schedstats=enable.`

Ensure all CPU vulnerabilities are mitigated

Profile Applicability: Level 1

Description: Ensure that all known CPU vulnerabilities are mitigated. This ensures that known attacks, such as Spectre, Meltdown, etc. either do not affect the node or are mitigated via software.

Rationale: Vulnerability mitigation ensures that known attacks cannot be exploited.

Impact: Impact analysis should be performed on a per-mitigation basis..

Audit: View CPU mitigations:

```
$ ls /sys/devices/system/cpu/vulnerabilities
```

Output may resemble the following:

```
  "Not affected"    CPU is not affected by the vulnerability
  "Vulnerable"      CPU is affected and no mitigation in effect
  "Mitigation: $M"  CPU is affected and mitigation $M is in effect
```

Remediation: Add the appropriate kernel arguments to the MachineConfigPools of the workers nodes and reboot them. See Documentation Chapter 1. Customizing nodes | Red Hat Product Documentation

Default Value: The cpu vulnerabilities file is not populated by default