**ocp4-cis-api-server-audit-log-maxsize,FAIL,medium,,  
" Run the following command**

$ oc get configmap config -n openshift-kube-apiserver -ojson | jq -r '.data["config.yaml"]' | jq '.apiServerArguments["audit-log-maxsize"]'

The output should return a value of ["100"] or as appropriate ComplianceCheckResult"  
Result: 200

**ocp4-cis-audit-log-forwarding-enabled,FAIL,medium,," Run the following command**

oc get clusterlogforwarders instance -n openshift-logging -ojson | jq -r '.spec.pipelines[].inputRefs | contains(["audit"])'

The output should return true.

ComplianceCheckResult"  
Result: False (No Instances are running)

**ocp4-cis-configure-network-policies-namespaces,FAIL,high,," Verify that the every non-control plane namespace has an appropriate**

**NetworkPolicy.**

To get all the non-control plane namespaces, you can do the

following command oc get namespaces -o json | jq '[.items[] | select((.metadata.name | startswith("openshift") | not) and (.metadata.name | startswith("kube-") | not) and .metadata.name != "default") | .metadata.name ]'

To get all the non-control plane namespaces with a NetworkPolicy, you can do the

following command oc get --all-namespaces networkpolicies -o json | jq '[.items[] | select((.metadata.namespace | startswith("openshift") | not) and (.metadata.namespace | startswith("kube-") | not) and .metadata.namespace != "default") | .metadata.namespace] | unique'

Make sure that the namespaces displayed in the commands of the commands match.

ComplianceCheckResult"  
Result: There are namespaces without Network Policy

**ocp4-cis-controller-rotate-kubelet-server-certs**

$ oc get configmaps config -n openshift-kube-controller-manager -ojson | jq -r '.data["config.yaml"]' | jq -r '.extendedArguments["feature-gates"]'

**The output should return RotateKubeletServerCertificate=true.**  
Result: Output doesn’t contain RotateKubeletServerCertificate=true

**ocp4-cis-kubelet-configure-event-creation**

Instructions: Run the following command on the kubelet node(s):

$ for NODE\_NAME in $(oc get nodes -ojsonpath='{.items[\*].metadata.name}'); do oc get --raw /api/v1/nodes/$NODE\_NAME/proxy/configz | jq '.kubeletconfig|.kind="KubeletConfiguration"|.apiVersion="kubelet.config.k8s.io/v1beta1"' | grep eventRecordQPS; done

**The output should return .**

Result: "eventRecordQPS": 50

**ocp4-cis-kubelet-enable-server-cert-rotation,FAIL,medium,," Run the following command on the kubelet node(s)**

$ for NODE\_NAME in $(oc get nodes -ojsonpath='{.items[\*].metadata.name}'); do oc get --raw /api/v1/nodes/$NODE\_NAME/proxy/configz | jq '.kubeletconfig|.kind="KubeletConfiguration"|.apiVersion="kubelet.config.k8s.io/v1beta1"' | grep RotateKubeletServerCertificate; done

The output should return true.

ComplianceCheckResult"

Result: No output

**ocp4-cis-node-master-kubelet-enable-protect-kernel-sysctl,FAIL,medium,," Run the following command on the kubelet node to check if sysctl configuration file exist(s)**

$ sudo [ -f /etc/sysctl.d/90-kubelet.conf ] || echo Not Exists

The output should not return Not Exists.

$ sudo grep vm.panic\_on\_oom /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.keys.root\_maxbytes /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.keys.root\_maxkeys /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.panic /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.panic\_on\_oops /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep vm.overcommit\_memory /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.panic /etc/sysctl.d/90-kubelet.conf

The output should return a value.

ComplianceCheckResult"  
  
Result: 90-kubelet.conf doesn’t exist, but 99-kubelet.conf exists

**ocp4-cis-node-worker-kubelet-enable-protect-kernel-sysctl,FAIL,medium,," Run the following command on the kubelet node to check if sysctl configuration file exist(s)**

$ sudo [ -f /etc/sysctl.d/90-kubelet.conf ] || echo Not Exists

The output should not return Not Exists.

$ sudo grep vm.panic\_on\_oom /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.keys.root\_maxbytes /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.keys.root\_maxkeys /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.panic /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.panic\_on\_oops /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep vm.overcommit\_memory /etc/sysctl.d/90-kubelet.conf

The output should return a value.

$ sudo grep kernel.panic /etc/sysctl.d/90-kubelet.conf

The output should return a value.  
Result: 90-kubelet.conf doesn’t exist, but 99-kubelet.conf exists

**ocp4-cis-ocp-api-server-audit-log-maxsize,FAIL,medium,," Run the following command**

$ oc get configmap config -n openshift-apiserver -ojson | jq -r '.data["config.yaml"]' | jq '.apiServerArguments["audit-log-maxsize"]'

The output should return a value of ["100"] or as appropriate.

ComplianceCheckResult"  
Result: 100

**ocp4-cis-scc-limit-container-allowed-capabilities,FAIL,medium,," Inspect each SCC returned from running the following command**

$ oc get scc

$ oc describe roles --all-namespaces

$ oc describe clusterroles

For any role/clusterrole that reference the

securitycontextconstraints resource with the resourceNames

of the SCCs that do not list an explicit allowedCapabilities, examine the

associated rolebindings to account for the users that are bound to the role.

Review each SCC and determine that only required capabilities are either

completely added as a list entry under allowedCapabilities,

or that all the un-required capabilities are dropped for containers and SCCs.

varible var\_sccs\_with\_allowed\_capabilities\_regex can be set to exclude certain

SCCs from the check.

Use following command to verify if the correct regex is being used, this ouput

$ oc get scc -o json | [.items[] | select(.metadata.name | test("^privileged$|^hostnetwork-v2$|^restricted-v2$|^nonroot-v2$"; "") | not) | select(.allowedCapabilities != null) | .metadata.name]

^privileged$|^hostnetwork-v2$|^restricted-v2$|^nonroot-v2$ should be replace to the actual value set,

either the default one or the one set from TailoredProfile.

ComplianceCheckResult"

Getting unqualified SCCs...

sukesh-sysdig-agent

sukesh-sysdig-nodeanalyzer

sysdig-agent

sysdig-agent-nodeanalyzer

trident-node-linux