NeuVector Onboarding and Best Practices Guide

INTRODUCTION 1

PLANNING 2

ARCHITECTURE 3
SIZING AND SCALE 4

DEPLOYMENT AND INITIAL CONFIGURATION 6

DEPLOYMENT TOOLS AND PRE-DEPLOYMENT CHECKLIST 6
MANAGING NEUVECTOR 7
POLICY MODES – DISCOVER, MONITOR, PROTECT 8
BACKUPS AND PERSISTENT DATA 9
INTEGRATION INTO CI/CD PIPELINE 10
ENTERPRISE INTEGRATION: ALERTS, NOTIFICATIONS, SIEM/SYSLOG, WEBHOOKS 10
POLICY MIGRATION – STAGING TO PRODUCTION 11
DOCKER NATIVE AND DOCKER EE / SWARM 12

OPERATIONS 13

VULNERABILITY AND COMPLIANCE MANAGEMENT 13
COMPLIANCE MANAGEMENT 14
IMPROVING THE SECURITY RISK SCORE IN THE DASHBOARD 14
NETWORK RULES – INGRESS, EGRESS, SEGMENTATION 15
AUTOMATION – REST API 16
AUTOMATION – SECURITY POLICY, CRD 17
REVIEWING NOTIFICATIONS AND REDUCING FALSE POSITIVES 18
UPDATING – NEUVECTOR, NODES, HOST OS, ORCHESTRATOR PLATFORMS 18

APPENDIX - PRE-DEPLOYMENT CHECKLIST 20

Introduction

This guide provides advice for planning and managing NeuVector deployments. References are provided to documentation for the ‘how to’ where possible.

For additional FAQ’s with best practices, be sure to ask your NeuVector representative for the NeuVector Customer FAQ document as well.

The majority of this guide focuses on Kubernetes or Kubernetes-based deployments such as Rancher and OpenShift. For deployment tips on docker-native hosts or Docker EE with Swarm, see the last section.

Throughout, general administrative knowledge of your particular deployment architecture is assumed. No two environments are alike. As this is an ever-evolving suite of technologies, be aware that some advice contained herein may not absolutely apply to you, and/or data may have changed since the last update of this document.

Planning

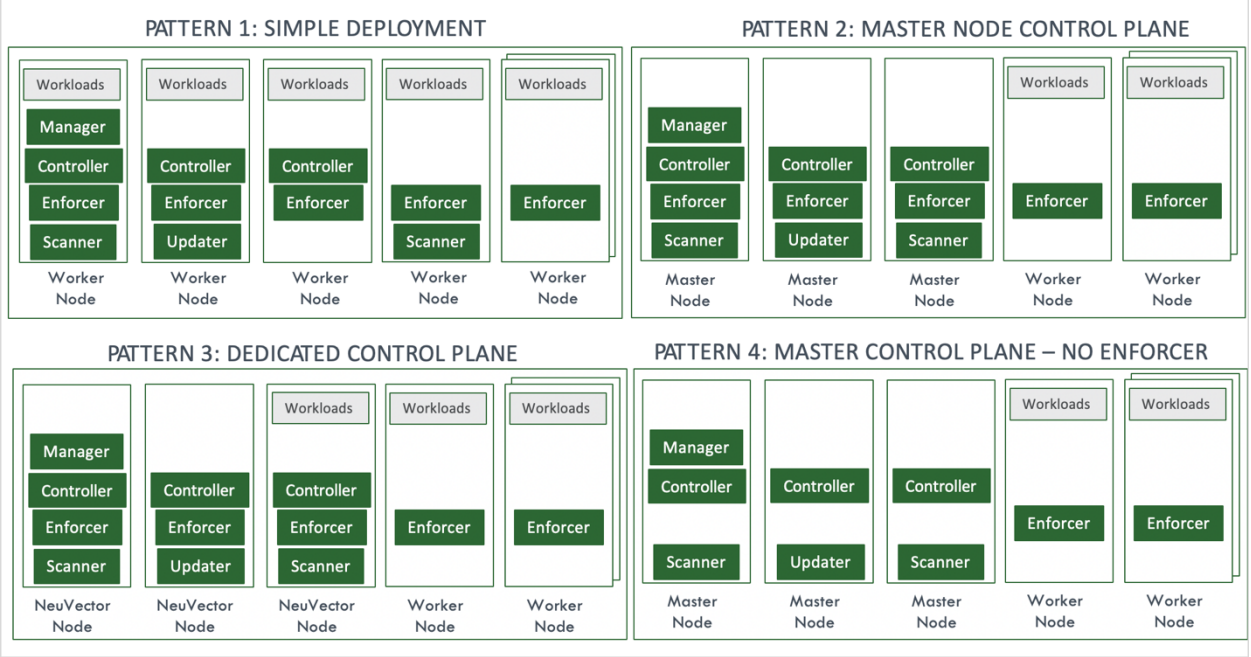
The NeuVector container security platform consists of several containers, all of which communicate to each on various ports and interfaces. Each container type should be evaluated to determine where to put them. Sizing and performance considerations are covered in the next section.

- **Manager**. A stateless container which presents the web-based console. Typically, only one is needed and this can run anywhere. Failure of the Manager does not affect any of the operations of the controller or enforcer. However, some notifications (events) and recent connection data are cached in memory by the manager so viewing of these would be affected.
- **Controller**. The 'control plane' for NeuVector, should be deployed in an HA configuration so configuration is not lost in a node failure. These can run anywhere, although in many cases customers choose to place these on 'management', master or infra nodes because of their criticality.
- **Enforcer**. This container is deployed as a daemonset so one Enforcer is on every node to be protected. Typically deploys to every worker node but scheduling can be enabled for master and infra nodes to deploy there as well. Note: If the Enforcer is not on a node in the cluster and connections come from a pod on that node, these are displayed as 'unmanaged' workloads in NeuVector.
- **Scanner**. Performs the vulnerability scanning using the built-in CVE database, as directed by the Controller. Multiple scanners can be deployed to increase scanning capacity. Scanners can run anywhere but are often run on the nodes where the controllers run. See below for sizing considerations of scanner nodes. A scanner can also be invoked independently when used for build-phase scanning such in a pipeline that triggers a scan, retrieves results, and stops the scanner. The scanner contains the latest CVE database so should be updated daily.
- **Updater**. The updater triggers an update of the scanner through a Kubernetes cron job when an update of the CVE database is desired. Please be sure to configure this for your environment.

Architecture

The simplest deployment pattern would be to let Kubernetes or the orchestrator determine where to put each container, based on the memory (and potentially cpu) request for each container. For clusters where each node is similarly resourced and adequate headroom for all workloads can be guaranteed this would be fine. Other considerations such as separate node maintenance cycles, network isolation, and sizing could affect where to place the NeuVector containers (except the Enforcer, which runs on every protected node).

The diagrams below show sample deployment patterns and the flexibility for NeuVector deployment.



1. Deployment where Kubernetes can place the NeuVector containers on any node, in this case worker nodes, unless scheduling to the Master has also been enabled. In a public cloud managed service like EKS, AKS, etc NeuVector can only run on worker nodes. The most common deployment.
2. NeuVector control plane containers and scanners are placed on the Master nodes through taints/tolerations or labels. Master nodes can be appropriately sized for control plane requirements.
3. Similar to 2, but nodes are selected to run the NeuVector control plane and scanners. These could be worker nodes and are appropriately sized. Note the one NeuVector node has workloads allowed to run on it as well, where the other don't allow workloads.
4. NeuVector control plane and scanners run on Master or dedicated nodes, but since there are no workloads running on them no Enforcer is deployed to them. In this case the system containers running on the master node are not monitored by NeuVector, similar to a public cloud case.

> ***Tip**: See Appendix A for a pre-deployment checklist. Deployment configuration or yaml files (from our sample) may need to be customized for your environment.*

Q&A

- Q: Can two or more Controllers run on the same node?
  A: Yes, although if the node goes down, all Controllers on that node would be lost. The sample deployment yaml has an 'affinity' setting which attempts to deploy controllers on separate nodes.
- Q: Should we deploy the Enforcer on the master and infra nodes?
  A: Yes, this is recommended if possible so it can monitor containers and network traffic on those as well.

References

- Preparation: https://open-docs.neuvector.com/basics/installation
- Deployment and sample yamls: https://open-docs.neuvector.com/deploying/production
- Updater for CVE database: https://open-docs.neuvector.com/scanning/updating

Sizing and Scale

The NeuVector containers require adequate memory and cpu to function properly. This is probably the most important consideration when planning the deployment architecture above. Make sure the minimum requirements (typically 1GB RAM) for each type of NeuVector container is allocated and available. This should be increased under certain conditions, such as:

- Large image scanning. The Scanner container must have enough memory to pull the image to be scanned into memory and expand it. If images larger than 1GB are to be scanned, increase memory to the scanner to slightly higher than the largest expected image size.
- High network connections expected in Protect mode. The Enforcer requires CPU and memory when in Protect (inline firewall blocking) mode to hold and inspect connections and possible payload (DLP). Increasing memory and dedicating a CPU core to the Enforcer can ensure adequate packet filtering capacity.

> ***Tip**: Configuring insufficient pod resource constraints on NeuVector containers can result in unexpected behavior.. We recommend that you **don't** place any memory or CPU constraints (maximums) on the NeuVector pods, or if required, make sure there is enough available headroom based on actual performance characterization in staging/production environments (with target workloads at scale).*

Other Scaling Considerations

- Number of Nodes in a Cluster. As the number of nodes increases in a cluster, the network communication from each enforcer to the controllers will collectively increase. Also, the practical management of viewing nodes in the web console may become more challenging.

- Number of Pods (Workloads) on a Node. The number of pods on a node will affect resource consumption and potentially network traffic (pod to pod on the node or to other nodes). There will also be more pods for the NeuVector Enforcer to gather scan data and compliance data to send to the Controller.

- Number of Namespaces, Containers and Other Assets in a Cluster. As containers, namespaces, groups, and other assets increase various displays in the console will become busier and may require more time to load. There are filters available in most screens to quickly isolate a namespace or container/pod to be viewed.

- Number of Images in a Registry to be Scanned. Consider deploying multiple scanner pods across different nodes if the number of images in a registry exceeds one thousand, and/or the time it is taking to scan or re-scan the entire registry/repository is longer than desired. Keep in mind each scanner pod will consume resources on its host during scanning. The auto-scaling feature can be set to automatically scale the number of scanner pods available up and down to meet demands.

- Number of Clusters in a Federation. The multi-cluster Primary initiates two-way connections between it and remote clusters. In addition, if federated rules are deployed these will be pushed to all remote clusters upon any change.

> ***Tip:** Observe the memory and cpu consumption statistics of all pods, including the NeuVector ones, in an environment similar to the expected production environment to make sure no adverse effects are observed.*

> ***Tip**: In the Network Activity map, select one or more namespaces and the checkmark to limit the view to selected objects. This will enable the map to load faster.*

Q&A

- Q: What are the minimum host specs for running the Controller?
  A: We recommend a minimum of 16 GB RAM and 4 CPU cores, assuming the Controller will be running with other system containers or workloads.
- Q: What is the maximum number of nodes in a cluster NeuVector can support?
  A: There is no hard limit placed by NeuVector, however, there are practical difficulties that may be experienced in clusters exceeding 500 nodes, depending upon the environment and host resources.

References

- System requirements, performance and scaling: https://open-docs.neuvector.com/basics/requirements

Deployment and Initial Configuration

Deployment Tools and Pre-Deployment Checklist

NeuVector supports several tools for deployment, including Helm charts, Operators, ConfigMaps and plain 'kubectl' yaml files.

> ***Tip:** Review the sample yaml files and/or Helm/Operator configuration options carefully and prepare a list of modifications for your own environment.*

Q&A:

- Q: What changes need to be made to the sample deployment defaults?
  A: Changes could include:
  - Manager service access method. Loadbalancer, nodeport, ingress controllers etc.
  - Image path's/registries/names/version #. If pulling renamed or retagged images from different registry.
  - Container run-time volume mounts. Default containerd, with CRI-O, docker and other run-times requiring changes to volume mounts.
  - Multi-cluster primary/remote. If multi-cluster is desired, enable service access.
  - Controller, scanner replicaset quantities. Default is 3 for controllers.

- Taints/tolerations for controlling node deployment. To control where controllers are deployed or if enforcers should be deployed on masters.

References

- Helm chart on Github: https://github.com/neuvector/neuvector-helm/tree/master/charts/core
- OpenShift Certified Operator: https://catalog.redhat.com/software/operators/detail/5ec3fa84ef29fd35586d9a16
- Community Operator: https://github.com/redhat-openshift-ecosystem/community-operators-prod/tree/main/operators/neuvector-community-operator
- Sample Kubernetes deployment: https://open-docs.neuvector.com/deploying/kubernetes

Managing NeuVector

Most NeuVector deployments are managed at least in part through the web-based console. However, the REST API, CLI, configMaps, and CRDs can all be used for non-console-based management.

> ***Tip**: Increase the Session Timeout so you don't get logged out after 5 minutes, in the My Profile menu (upper right).*

Simple deployments can use an integrated load balancer (such as on EKS, AKS, GKE, IKS) to enable external access to the Manager service, or expose a nodePort for access. Note that there are security considerations for nodePort access, as it exposes the port on every node for external access rather than forcing ingress through a load balancer or ingress controller.

> ***Tip**: Use a load balancer or ingress controller (e.g. nginx) to control access to the NeuVector console.*

> ***Tip**: The SSL connection can be terminated at the ingress controller and HTTP used to the manager. Use the environment variable in the Manager deployment to turn off SSL to the manager.*

Q&A

- Q: Can the Manager container run outside the cluster?
  A: Yes, although this is unusual. The Manager requires that the REST API to the controller be exposed outside of the cluster.

- Q: Can a service mesh ingress controller such as Istio be used for the NeuVector manager service?
  A: No, this is not supported. Use an alternative Kubernetes ingress method.
- Q: Can the Manager manage multiple clusters?
  A: Yes, this requires deployment of a Multi-cluster federation of controllers. A separate license may also be required.
- Q: Can the self-signed certificates be replaced?
  A: Yes, see the reference links below for documentation.

References

- Connect to Manager, REST API: https://open-docs.neuvector.com/configuration/console
- Replace Certificate for Manager: https://open-docs.neuvector.com/configuration/console/replacecert
- Replace Internal Certificates: https://open-docs.neuvector.com/deploying/production/internal

Policy Modes – Discover, Monitor, Protect

The policy mode of each Group determines what NeuVector does when it detects processes, network connections, and file activity. In general, Discover mode should be used in testing and staging environments to build the whitelist rules which protect the application workloads. Monitor or Protect mode should be used in Production environments to respond to security events. Monitor or Protect mode can also be used in test/staging environments to observe the run-time behavior expected of NeuVector in production environments.

> ***Tip**: After deploying workloads in Discover mode, run test scripts or traffic that will exercise all functions in the container (network connections, process, file). When new rules have not been created for a few days, switch the Group to Monitor mode and observe for at least a week. Look for Notification -> Security Events to see if legitimate activity is being alerted. Whitelist legitimate activity by adding the appropriate network or process rules. If Protect (blocking) mode is desired, switch the Group to Protect mode and pay special attention during the next few days to any blocked activity.*

> ***Tip**: Zero-drift mode is enabled by default for process and file protections in containers. This is useful for hardened containers where limited functions/processes is allowed.*

Q&A

- Q: How long should I leave the group in Discover mode before moving to Monitor or Protect?
  A: This could be as fast as a few hours, after all application tests have been run and you are confident all network connections and processes have been learned by NeuVector. Or it could take days or a week. For example, some open source tools make periodic connections externally to check for updates, and you may not see these. You should decide if these external connections should be allowed, and if so, whitelist rules added for them. A good practice is to export the rules as a CRD yaml file and review them with the application developers to confirm the expected behavior.

References

- Policy Modes: https://open-docs.neuvector.com/policy/modes
- Zero-drift: https://open-docs.neuvector.com/policy/processrules#zero-drift-process-protection

Backups and Persistent Data

The NeuVector configuration as well as any state data (connections, notifications etc) are sync'd between available controllers. However, if all controllers go down, the configuration and state data will be lost. To enable NeuVector to automatically recover the configuration of the cluster after an outage, enable a persistent volume. When the controller(s) start, they will pull the latest backed up configuration from the persistent volume.

> ***Tip**: Create a RWX persistent volume to automatically backup the NeuVector configuration, and take manual snapshots through the Console or REST API regularly and before any NeuVector, host OS, or orchestrator updates (reboots). Some public cloud storage systems don't support RWX so separate storage such as NFS may need to be deployed.*

Q&A

- Q: Can the exported backup file be imported into a different cluster to configure it?
  A: This is not recommended, as there may be changes in names, namespaces, IP addresses, system containers or other configuration settings that don't work in the target cluster. Use Helm, configMaps and CRDs to automate configuration of clusters.

References

- Configuring Persistent Volume: https://open-docs.neuvector.com/deploying/production#backups-and-persistent-data

Integration Into CI/CD Pipeline

Security should be as integrated and automated into the CI/CD pipeline as possible. Vulnerability and compliance management is covered in the next section. Integration can be done with the plug-ins and supported interfaces in NeuVector, or customized using the REST API (See Automation – REST API section). Admission control is a critical bridge between the pipeline and the production environment and is recommended to be enabled.

> ***Tip**: Enable and test the admission controller in Policy -> Admission Control. Then create a few simple rules to block unauthorized deployments. Even if registry scanning has not been configured yet, rules based on registry names, namespaces or other general criteria can place safeguards around image deployments.*

References

- REST API documentation: https://open-docs.neuvector.com/automation/automation
- NeuVector github including CircleCI Orb, Bamboo: https://github.com/neuvector

Enterprise Integration: Alerts, Notifications, SIEM/SYSLOG, Webhooks

NeuVector displays alerts in the Notifications menu and exports events via SYSLOG, webhooks, or a Prometheus exporter. Events can also be exported using the REST API. The most recent events for each type of event (security events, risk reports, and general events) are displayed in NeuVector. However, these are limited to the most recent 4k events of each type. It is expected that events will be exported via SYSLOG or other means for permanent storage, alerting and advanced processing.

> ***Tip**: Use webhooks to send special event notifications directly to a webhook endpoint (e.g. Slack) or to a custom webhook receptor container within your cluster for additional processing.*

> ***Tip**: For custom integration with alerting/paging systems, case management systems, or SIEM use the REST API, webhooks, or a combination.*

Policy Migration – Staging to Production

Once applications are tested and the NeuVector run-time security rules verified, the rules should be replicated in the production environment.

> ***Tip**: Use the NeuVector CRD to export, review, check-in, and migrate security rules from a staging to production environment.*

> ***Tip**: In Production, set the New Services Mode to Monitor or Protect in Settings -> Configuration to prevent any unknown services from starting without generating alerts. Before deploying any new workloads, make sure the whitelist rules (process, network, file) are deployed through a CRD, REST API, or console so the workloads can start running without interruption in a protected state.*

Q&A

- Q: How long should we run in staging before moving to production?
  A: Make sure your staging environment has the same nodes, orchestrator, system-containers and other important assets as in production. Initial application workloads expected in production should also be tested, but expect new and updated applications to occur continuously. This is typically a few weeks in staging for initial deployments, and days to weeks for additional or updated application workloads.
- Q: Should we run in Monitor or Protect mode in production?
  A: Initially, we recommend you run all groups in Monitor mode in production for a few days or weeks until you are comfortable with any security events detected in Notifications. Then, you can switch to Protect mode for only those groups that you wish NeuVector to block network, process, and file violations. This can be decided based on groups with egress connections, or critical databases, or workloads that you are 100% sure that all expected behavior has been whitelisted.
- Q: Can I use the export/import configuration file in Settings -> Configuration for the migration?
  A: This is not recommended, as there may be changes in names, namespaces, IP addresses, system containers or other configuration settings that don't work in the target cluster. Use CRDs to do the migration by exporting Group rules and editing them in the yamls to reflect any changes in the production environment before deploying them. ConfigMaps can be used for consistent configuration of other settings in both staging and production environments.

References

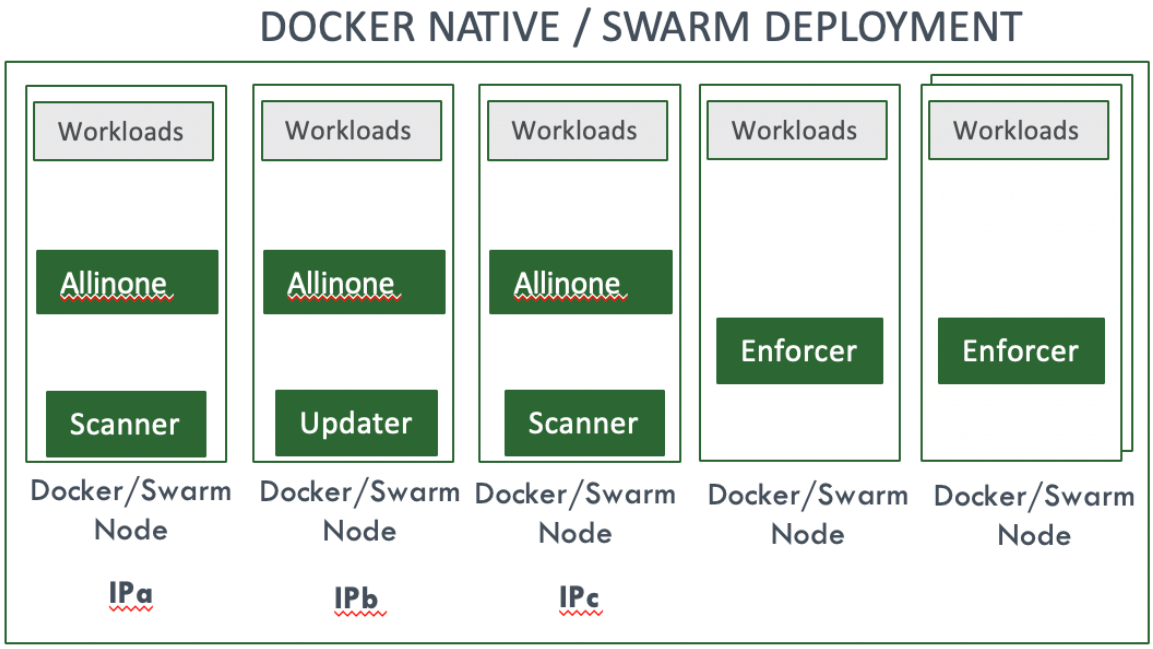
- Using CRD: https://open-docs.neuvector.com/policy/usingcrd

Docker Native and Docker EE / Swarm

Deployments and features are very different for docker-native or Swarm. Many NeuVector features leverage Kubernetes or Openshift resources and therefore are not possible in docker-native or Swarm. The features NOT supported in this environment include (but not limited to):

- Admission control
- CRD (policy as code)
- Rolling updates
- Helm, Operator based deployments
- ConfigMaps
- Some automated classification of 'system' containers
- Persistent volumes.

Deployment on individual docker hosts or Swarm nodes is very different than Kubernetes. The recommended deployment pattern is to deploy the NeuVector Allinone container on the first 3 nodes, then the Enforcer on any node after that. The 'cluster' will be formed by connecting all the Allinone's together and the Enforcers connected to the Allinone cluster.



The Allinone containers the Manager, Controller, Enforcer all in one container. This provides for console access from any of the Allinone nodes as well as HA from the three controllers. When deploying using the docker run or docker-compose commands, the CLUSTER_JOIN_ADDR environment variable is used to form the cluster of the Allinones. Use the IP addresses of the three Allinone nodes separated by comma's when deploying each Allinone and Enforcer. For example, CLUSTER_JOIN_ADDR=IPa,IPb,IPc

Q&A

- Q: What if I only have one or two nodes?
  A: The Controller function in the Allione requires an odd number to elect a leader, so deploy a single Allinone and an Enforcer for two nodes, or a single Allinone for one node. Automated HA is not possible in these configurations. Be sure to backup the configuration to handle outages.

References

- Docker deployment: https://open-docs.neuvector.com/deploying/docker#deploy-neuvector-containers-using-docker-native-or-ucpswarm

Operations

Vulnerability and Compliance Management

Every company will have a different process and standards for managing vulnerabilities and compliance tests. NeuVector has the flexibility to adapt to your process.

Key best practices for vulnerability and compliance management include:

- Require developers to remediate critical vulnerabilities if a fix is available, stopping or alerting as early as in the build phase if possible.
- Notify developers or appropriate teams if a new vulnerability is discovered in an existing (approved) image in a registry, or in a production container.
- Provide a grace period to allow developers to remediate critical vulnerabilities, but ensure that running workloads are protected by NeuVector whitelist rules and 'virtual patching.' See the Q&A and reference link below for virtual patching, which protects containers running with vulnerabilities.
- Allow developers and devops teams to apply for exceptions to policy, and be able to ignore certain vulnerability scan alerts (based on CVE numbers).

> ***Tip**: Utilize the fields "With Fix," "Published date," and custom "Author/developer" metadata to automate policies that require a developer to fix vulnerabilities (with fix available) that have been published more than 7 days ago (a grace period).*

Q&A

- Q: Why does the NeuVector scan report differ from another scanner I'm using?
  A: Each scanning vendor maintains its own CVE database as well as the interpretation of the CVE sources such as severity/criticality levels.
- Q: Can NeuVector scan an image as soon as it is pushed to a registry?
  A: Some registries such as Openshift support imagestreams which enable NeuVector to

scan an image automatically when pushed. For others, a periodic scan can be configured to scan new images as often as every few minutes to hours. For true on-demand scanning, the REST API can be used to trigger a scan on a particular image after it is pushed to a registry.

- Q: What is Virtual Patching?
  A: This term when used by NeuVector means that a container running in production with critical vulnerabilities is 'virtually patched' by NeuVector when running in Monitor or Protect mode, because any attempt to exploit the vulnerability will be immediately detected and blocked as it creates an unauthorized process, network connection, or file access.

References

- Documentation: https://open-docs.neuvector.com/scanning/scanning

Compliance Management

Compliance checks in NeuVector include both CIS benchmarks (docker, kubernetes, openshift etc) as well as custom compliance checks (scripts run in containers or on hosts). These can be tagged and reported on for various industry standards such as PCI, GDPR and others.

> ***Tip**: Use the CIS and Compliance results listed for PCI, GDPR etc or for each Asset (node, container) as the complete list of compliance checks for the asset(s) for reporting to auditors. Use the Security Risks -> Compliance menu for listing and prioritizing compliance violations that may need to be addressed by devops teams.*

Q&A

- Q: Can I customize the compliance reports?
  A: Yes, standard reports for PCI, GDPR and others can be created by tagging the appropriate compliance checks. Each of these can be customized so the reports include or exclude certain checks.

Improving the Security Risk Score in the Dashboard

The Dashboard provides an overall Security Risk Score which is based on the Policy mode of containers, ingress/egress connections, vulnerabilities, privileged/root containers, and admission control. Use the wizard tool next to the score to improve your score step by step.

> ***Tip**: It is often not possible to get the Risk Score to zero, because any running container, Kubernetes system container, or egress connection represents a risk. Any score in the Good range (less than 20) is considered acceptable.*

References

- How to improve Risk Score: https://open-docs.neuvector.com/navigation/improve_score

Network Rules – Ingress, Egress, Segmentation

Network segmentation, inspection and protection are some of the most critical security protections available during run-time. Network rules should be carefully reviewed and adjusted to achieve the desired behavior (allowing, alerting, blocking).

Network rules learned in Discover mode by NeuVector can become fragmented under certain conditions, such as:

- Connection source or destinations are always changing because random ports are used, for example if coming through a load balancer or ingress
- Pod deployments for the same application or new versions of the same application have version numbers or random strings in their naming conventions, causing NeuVector to think it's a new application and create new rules for it.

> ***Tip**: Review and edit Network rules affecting each application by filtering on the application in Policy -> Network Rules. If you notice many rules repeated with a From or To of changing IP addresses, ports or nodes, think about how a higher level network rule based on Protocol, Labels, or other criteria (with or without wildcards) could be created to consolidate such rules.*

Egress connections can be a source of high risk and should be evaluated and if possible, declared to allow access only to specific destinations.

> ***Tip**: Create custom egress rules for workloads requiring access outside the cluster. Create the target (destination) custom Group using 'address=<destination>" and a corresponding Network rule allowing access from a pod Group to the target Group, using an Application protocol (e.g. MySQL, redis, mongodb, SSL etc) if possible.*

Pod labels applied during deployment can also be used to enforce rules. For example, scope=cde, scope=non_cde, external_access=allowed are examples of labels which can be applied to pods, and rules created in NeuVector for those pods. A custom Group can be created which matches the label, and rules applied to that Group.

Q&A

- Q: Can network rules be ordered?
  A: Yes, custom and learned network rules can be ordered through the console or REST API. Federated rules and CRD created rules can't be edited, and always are evaluated first (ie, above other rules).
- Q: When creating a custom Group, can wildcards be used?
  A: Yes, wildcards are generally supported in the criteria, for example 'address=*.google.com'. If more flexible matching is desired a regex expression can also be used by indicating it with the ~ sign such as 'label~my.label*-xyz'.
- Q: Can I change the 'policymode' of a custom Group to Monitor or Protect?
  A: Currently the policy mode of a custom group is not configurable because the underlying pods which are referred to could be in different modes (Discover, Monitor, Protect).
- Q: Can the DLP network payload inspection regex engine be used for other policies?
  A: Yes, it can be used for secrets inspection, for inspecting HTTP headers or paths, or other purposes.
- Q: Can some Groups be set to block, while other set to just Alert?
  A: Yes, each learned Group in NeuVector can be in a different mode, Discover, Monitor or Protect.
- Q: Can a Group be set to block external egress connections, but only alert if there is a violation from another pod within the cluster or namespace?
  A: Not at this time. A Group can only be in one mode for all traffic. We are considering more granular policies for a future release.

References

- Network Rules: https://docs.neuvector.com:1594/policy/networkrules
- Egress Control in Kubernetes, OpenShift, Istio, NeuVector: https://neuvector.com/container-security/enforce-egress-control-containers/

Automation – REST API

Use the REST API to perform any actions directly on the Controller without having to go through the web console. The Manager container uses the REST API to access the Controller. The default port of the REST API is 10443, which can be accessed from within the cluster. To make calls to the controller from outside the cluster, expose the REST API as a service externally.

> ***Tip**: Use the REST API in scripts to automate things such as policy backup, packet capture based on suspicious activity, triggering image scans, pulling scan results.*

Q&A

- Q: How does authentication and authorization get enforced?
  A: The api token request requires a user and password for authentication. The role of the user determines what actions are allowed through the api for that user.
- Q: How long does the token last?
  A: The token lasts the same length as the user's timeout, set in the user's profile. For example, if the timeout is set to 10 minutes (600 seconds), the token will be valid as long as it is used within 10 minutes. After 10 minutes of inactivity it will expire.

References

- API documentation: https://open-docs.neuvector.com/automation/automation

Automation – Security Policy, CRD

The NeuVector Custom Resource Definition (CRD) provides a powerful mechanism for declaring and automating security rules in NeuVector. It can enable collaboration between security, devops and developers to discuss and review allowed application behavior.

> ***Tip**: Use NeuVector to learn application behavior in staging/test, then export the rules as a CRD to review and approve with the developer and devops teams. Then check-in the CRD to manage security policy like other 'code.'*

CRDs are used to 'declare' a set of Groups, rules, and the policy mode of NeuVector objects. Once declared, these can only be edited by applying an updated CRD.

> ***Tip**: Use CRDs to set 'global' security rules that are not tied to specific application behavior, such as preventing SSH or SCP on 'all containers' (containers is a reserved group name in NeuVector which can be used for this). Or allow external api access to pods with certain labels.*

Q&A

- Q: How do I export the CRD for my applications?
  A: Go to Groups and select the ones for export, then click Export Group Policy. All rules for select groups, AND related groups (ie, those defined in network rules as sources or destinations for a selected group) will be exported.
- Q: Can I change the 'policymode' of a custom Group before applying in a CRD?
  A: Currently the policy mode of a custom group must be 'null' because the underlying pods which are referred to could be in different modes (Discover, Monitor, Protect).

References
- CRD syntax: https://open-docs.neuvector.com/policy/usingcrd

Reviewing Notifications and Reducing False Positives

During the first few days or weeks of NeuVector deployment events should be reviewed to determine if they are false positives. There are several methods of reducing false positives:

1. Review the event in Notifications and if applicable, select the Review Rule button to immediately add a whitelist rule for the violation event.
2. After reviewing the event notification, create a whitelist rule manually in the appropriate Policy menu – admission control, process (under Group), network rules etc.
3. Create a Response rule to 'suppress notification' for those types of events.

> ***Tip**: Whenever possible, create whitelist rules to allow behavior that is being reported as a violation. For temporary suppression of notifications use a Response rule, which can easily be disabled or removed later.*

References
- Notifications: https://open-docs.neuvector.com/reporting/reporting
- Response Rules: https://open-docs.neuvector.com/policy/responserules
- NeuVector github including Prometheus exporter: https://github.com/neuvector

Updating – NeuVector, Nodes, Host OS, Orchestrator Platforms

NeuVector supports rolling updates of its critical containers, but special care should be taken for any updates of the environment. The controllers maintain a state between themselves and this will be lost if all controllers become unavailable at the same time. For this reason, take special care when upgrading the hosts/nodes (rebooting) or the orchestrator (e.g. Kubernetes) even if a node draining process is invoked.

> ***Tip**: For updates of host OS or the orchestrator platform such as Kubernetes which require node reboots or pod draining, make sure at least one NeuVector Controller is active at all times. When rebooting a node with a Controller, observe the new Controller to ensure it becomes available for at least 60 seconds (a few minutes is better) to make sure it has the time to sync state with the leader Controller, BEFORE rebooting the next node with a Controller.*

> ***Tip**: Always make a manual backup of the entire configuration before any update of NeuVector, the hosts it runs on, or the orchestrator. This can be exported in Settings -> Configuration -> Export All*

References

- Rolling updates: https://open-docs.neuvector.com/updating/updating
- Configuring Persistent Volume: https://open-docs.neuvector.com/deploying/production#backups-and-persistent-data
- Updating the CVE database: https://open-docs.neuvector.com/scanning/updating

Appendix - Pre-Deployment Checklist

- ❑ 1. Gather Required Information
  - ❑ NeuVector version # and target platform orchestrator version #.
  - ❑ Dockerhub ID for pulling NeuVector images.
  - ❑ Target nodes CPU/memory profiles.
  - ❑ Container run-time being used.
  - ❑ Method available for ingress to Manager.
  - ❑ Integration info for SYSLOG servers, LDAP/AD, SSO/SAML servers

- ❑ 2. Review NeuVector Documentation
  - ❑ https://open-docs.neuvector.com/, especially section 1 (Deployment Preparation) and section 2 (Deploying). Each orchestration platform also has specific deployment instructions in this section.

- ❑ 3. Prepare the Target Environment
  - ❑ Pre-pull images if not dynamically pulling from NeuVector docker hub registry.
  - ❑ Create RWX storage volume if using persistent storage for configuration backup

- ❑ 4. Ensure Connectivity
  - ❑ Test ability to pull images from within the cluster, from registry or docker hub.
  - ❑ Enable and test access to registry(s) from within the cluster for registry scanning if applicable.
  - ❑ Console network access to manager service in cluster through load balancer, route, IP/port (default port 8443).
  - ❑ Check to make sure network or local firewalls e.g. firewalld are not blocking access to required ports for NeuVector.
  - ❑ Enable outbound connections if required for SYSLOG (default port 514) and webhook notifications.

- ❑ 5. Create Deployment Process and Templates
  - ❑ Decide on deployment method: Helm, kubectl/yaml files, Operator…
  - ❑ Review sample yaml files and/or values and configuration options
    - ❑ Kubernetes cluster roles and security yaml examples
    - ❑ Deploying from Rancher Manager
    - ❑ Openshift Helm Chart or yaml Files
  - ❑ Edit yamls or deployment values in Helm as applicable:
    - ❑ Manager access - LoadBalancer, ingress, NodePort…
    - ❑ Enable/disable multi-cluster master and remote services
    - ❑ Image name, version tag, or path to NeuVector images

- ❑ Container run-time if containerd or CRI-O
- ❑ Taints/tolerations or node labels for controlling where NeuVector pods are deployed.