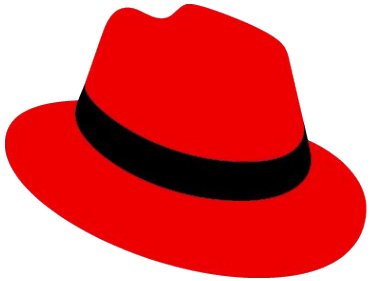


Red Hat

OCP 4.14 Stateless Ingress Node Firewall

Presenter
Mohamed S. Mahmoud

RED HAT® OPENSHIFT Container Platform

Red Hat

What does it do

Secure OpenShift nodes from external (DOS) attacks by configuring specific **stateless** policies

How does it work

- Ingress node firewall operator uses CR to deploy and configure ingress node firewall rules
- Webhook validates the configuration and avoids malformed configuration from locking up OCP cluster access (fail safe rules)
- XDP+eBPF apply ingress node firewall rules, parse packets, process rule actions, update statistics and generate syslog events for dropped packets

Webhook - Ingress firewall Fail Safe rules

- Prevents users from locking up cluster access
  - for example by denying TCP traffic to the API server port 6443
- Webhook verification refuses **any** rules that match against critical ports for TCP and UDP with **deny** action

Source CIDR	Protocol	Port	Purpose
0.0.0.0/0	TCP	22	SSH
0.0.0.0/0	UDP	68	DHCP
0.0.0.0/0	TCP	6443	Kubernetes API server access
0.0.0.0/0	TCP	2379/2380	Etcd access
0.0.0.0/0	TCP	10250	Kubelet
0.0.0.0/0	TCP	10259	Kube-scheduler
0.0.0.0/0	TCP	10257	Kube-controller-manager

Why XDP + eBPF

eBPF provides a very flexible mechanism to allow early detection and packet filtering

eXpress Data Path (XDP) is a feature in the Linux kernel which …

- allows users to execute a user-supplied eBPF program when a packet is received on a network interface (NIC)
  - **flexibility**
- allows attachment of an eBPF program at the earliest networking driver stage
  - **high performance**
- allows processing of accepted packets via the Linux kernel
  - **no need to reinvent the wheel**

How to deploy Ingress Node Firewall

Enable subscription

```
cat <<EOF | oc apply -f -
kind: Namespace
metadata:
  name: openshift-ingress-node-firewall
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ingress-node-firewall
  namespace: openshift-ingress-node-firewall
spec:
  upgradeStrategy: Default
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
  name: ingress-node-firewall-sub
  namespace: openshift-ingress-node-firewall
spec:
  channel: alpha
  name: ingress-node-firewall
  source: ingress-node-firewall
  sourceNamespace: openshift-ingress-node-firewall
EOF
```

Deploy IngressNodeFirewallConfig

```
cat <<EOF | oc apply -f -
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewallConfig
metadata:
 name: ingressnodefirewallconfig
 namespace: openshift-ingress-node-firewall
spec:
 nodeSelector:
   node-role.kubernetes.io/worker: ""
EOF
```

Check installed CRDs and running pods

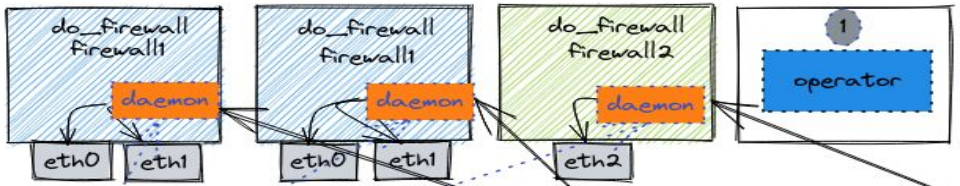
oc get crds | grep ingressnodefirewall

```
ingressnodefirewallconfigs.ingressnodefirewall.openshift.io      2022-08-25T10:03:01Z
ingressnodefirewallnodestates.ingressnodefirewall.openshift.io   2022-08-25T10:03:00Z
ingressnodefirewalls.ingressnodefirewall.openshift.io            2022-08-25T10:03:00Z
```

oc get pods -n openshift-ingress-node-firewall

```
NAME                                                  READY   STATUS    RESTARTS   AGE
ingress-node-firewall-controller-manager-656dc8dc7-cr8g5   2/2     Running   0          5d21h
ingress-node-firewall-daemon-pqx56                    3/3     Running   0          5d21h
```

Operator and configuration overview




```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewallConfig
metadata:
  name: ingressnodefirewallconfig
  namespace: ingress-node-firewall-system
spec:
  nodeSelector:
    do_firewall2: ""
```

2

```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewall
metadata:
  name: ingressnodefirewall-demo-1
spec:
  interfaces:
  - eth0
  - eth1
  nodeSelector:
    matchLabels:
      firewall1: 'true'
  ingress:
  - sourceCIDRs:
      - 1.1.1.1/24
      - 100:1::1/64
    rules:
    - order: 10
      protocolConfig:
        protocol: TCP
        tcp:
          ports: "100-200"
      action: Allow
    - order: 20
      protocolConfig:
        protocol: UDP
        udp:
          ports: 8000
      action: Allow
```

3

```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewall
metadata:
  name: ingressnodefirewall-demo2
spec:
  interfaces:
  - eth2
  nodeSelector:
    matchLabels:
      firewall2: 'true'
  ingress:
  - sourceCIDRs:
      - 172.16.0.0/12
    rules:
    - order: 10
      protocolConfig:
        protocol: ICMP
        icmp:
          icmpType: 8 #ICMP Echo request
      action: Deny
    - order: 20
      protocolConfig:
        protocol: TCP
        tcp:
          ports: "8000-9000"
      action: Deny
  - sourceCIDRs:
      - fc00:f853:ccd:e793::0/64
    rules:
    - order: 10
      protocolConfig:
        protocol: ICMPv6
        icmpv6:
          icmpType: 128 #ICMPV6 Echo request
      action: Deny
```

3

1-deploy ingress-node-firewall operator.
2-deploy ingress node firewall daemonset ONLY for nodes with matching nodeSelector.
3-apply ingress node firewall rules, you can select which subset of the nodes the rules will be applied on using matchLabels.
4-NodeState object(s) will be created on every node based on the number of interfaces in the firewall rules.
5-NodeState object will program eBPF rules and attach XDP programto the target interface.

Red Hat

Ingress Firewall rules (1/5)


```yaml
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewall
metadata:
 name: ingressnodefirewall-demo2
spec:
 interfaces:
 - eth0
 nodeSelector:
   matchLabels:
     do-node-ingress-firewall: 'true'
 ingress:
 - sourceCIDRs:
      - 172.16.0.0/12
   rules:
   - order: 10
     protocolConfig:
       protocol: ICMP
       icmp:
         icmpType: 8 #ICMP Echo request
     action: Deny
   - order: 20
     protocolConfig:
       protocol: TCP
       tcp:
         ports: "8000-9000"
     action: Deny
```

Ingress Firewall rules (2/5)


```
(...)
spec:
  interfaces:
  - eth0
  nodeSelector:
    matchLabels:
      do-node-ingress-firewall: 'true'
(...)
```

- Specify the interfaces that rules apply to
- Using nodeSelector matchLabels user can select which nodes to apply the firewall rules to

Ingress Firewall rules (3/5)


```
(...)
ingress:
 - sourceCIDRs:
     - 172.16.0.0/12
   rules:
   - order: 10
     protocolConfig:
       protocol: ICMP
       icmp:
         icmpType: 8 #ICMP Echo request
     action: Deny
   - order: 20
     protocolConfig:
       protocol: TCP
       tcp:
         ports: "8000-9000"
     action: Deny
```

- User can configure multiple CIDRs from different address-families up to 1000 different CIDRs.
- Ingress firewall rules are ordered starting from 1 for each sourceCIDR(s) with up to 100 rules per CIDR.

Ingress Firewall rules (4/5)


```
(...)
ingress:
 - sourceCIDRs:
      - 172.16.0.0/12
   rules:
   - order: 10
     protocolConfig:
       protocol: ICMP
       icmp:
         icmpType: 8 #ICMP Echo request
     action: Deny
   - order: 20
     protocolConfig:
       protocol: TCP
       tcp:
         ports: "8000-9000"
     action: Deny
```

- Supported protocols are TCP, UDP, SCTP, ICMP and ICMPv6
- For ICMP and ICMPv6 rules can match against ICMP/ICMPv6 type and/or code.
- Ingress firewall actions are either **Allow** or **Deny**
- For protocols TCP, UDP and SCTP rule can match against single DstPort or range of ports using “start-end” format

Ingress Firewall rules (5/5)

- Matching filters are optional and ignored if they are not specified
- Ingress firewall rules get verified using verification webhook to fail any invalid configuration
- Allow and Deny packets and bytes count statistics is available per rule
- For dropped packets by XDP program an event will be emitted to syslog which includes event header (ruleId, Interface the packet came in on, packet length in bytes including L2 header) and up to 256 bytes from the packet header

Zero Trust Ingress Firewall rules


```yaml
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewall
metadata:
 name: ingressnodefirewall-zero-trust
spec:
 interfaces:
 - eth1
 nodeSelector:
   matchLabels:
     do-node-ingress-firewall: 'true'
 ingress:
 - sourceCIDRs:
     - 0.0.0.0/0
   rules:
   - order: 10
     protocolConfig:
       protocol: TCP
       tcp:
         ports: 22
     action: Allow
   - order: 20
     action: Deny
```

Match any CIDR

Normally used with Multi Interfaces clusters.
Where user wanted to drop all traffic on specific interface except maybe SSH for additional security.

Red Hat

Red Hat OpenShift

Administrator
- Home
- Operators
  - OperatorHub
  - Installed Operators
- Workloads
- Networking
- Storage
- Builds
- Observe
- Compute
- User Management
- Administration

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: All Projects

OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through Red Hat Marketplace. You can install Operators on your clusters to provide optional add-ons a your developers. After installation, the Operator capabilities will appear in the Developer Catalog providing a self-service experience.

All Items
- AI/Machine Learning
- Application Runtime
- Big Data
- Cloud Provider
- Database
- Developer Tools
- Development Tools
- Drivers and plugins
- Integration & Delivery
- Logging & Tracing
- Modernization & Migration
- Monitoring
- Networking
- OpenShift Optional
- Security
- Storage
- Streaming & Messaging
- Other

Source
- Red Hat (1)
- Certified (7)
- Community (0)
- Marketplace (2)
- Production Operators (2)

Provider
- Red Hat (2)
- APIMatic.io (0)
- Aerospike (0)
- Alvearie (0)

All Items

ingress

Production Operators
AWS Load Balancer Operator
provided by Red Hat
Operator to simplify management of aws-load-balancer-controller ### Prerequisites for installatio...

Red Hat
AWS Load Balancer Operator
provided by Red Hat
Operator to simplify management of aws-load-balancer-controller ### Prerequisites for installatio...

Certified
Citrix ADC CPX Istio Sidecar Injector Operator
provided by citrix
In Istio service mesh, a sidecar proxy runs alongside application pods and it intercepts and...

Marketplace
Citrix ADC CPX with Ingress Controller
provided by Citrix
Citrix ADC CPX is a container-based application delivery controller that can be provisione...

Certified
Citrix ADC CPX with Ingress Controller
provided by Citrix
Citrix ADC CPX is a container-based application delivery controller that can be provisione...

Certified
Citrix ADC Istio Ingress Gateway Operator
provided by citrix
An Istio ingress gateway acts as an entry point for the incoming traffic. Citrix ADC CPX, MPX, or...

Certified
Citrix Ingress Controller
provided by Citrix
Citrix provides an ingress controller for Citrix ADC MPX (hardware), Citrix ADC VPX...

Marketplace
Citrix Ingress Controller
provided by Citrix
Citrix provides an ingress controller for Citrix ADC MPX (hardware), Citrix ADC VPX...

Certified
IBM Security Verify Operator
provided by IBM
The IBM Security Verify operator can consistently enforce policy-driven security, including...

Production Operators
Ingress Node Firewall Operator
provided by RedHat
The Ingress node firewall...
Installed

Certified
Kong
provided by Kong Inc.
Kong Operator for Kubernetes. Technical preview.

Certified
Nginx Ingress Operator
provided by NGINX Inc
The NGINX Ingress Operator is a Kubernetes/OpenShift component which deploys and...

Red Hat

Administrator
- Home
- Operators
  - OperatorHub
  - Installed Operators
- Workloads
- Networking
- Storage
- Builds
- Observe
- Compute
- User Management
- Administration

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: openshift-ingress-node-firewall

Installed Operators > Operator details

Ingress Node Firewall Operator
4.12.0-202212061458 provided by RedHat

Actions

Details | YAML | Subscription | Events | All instances | IngressNodeFirewallConfig | IngressNodeFirewallNodeState | IngressNodeFirewall

Provided APIs

INFC IngressNodeFirewallConfig
Not available
Create instance

INFN IngressNodeFirewallNodeState
Not available
Create instance

INF IngressNodeFirewall
Not available
Create instance

Provider
RedHat

Created at
Dec 13, 2022, 10:09 AM

Links
Ingress Node Firewall
https://github.com/openshift/ingress-node-firewall

Maintainers
msherif1234
mmahmoud@redhat.com

Description

The Ingress node firewall operator, manages the configuration of cluster scoped Ingress Node firewall rules for Kubernetes.

ClusterServiceVersion details

Name
ingress-node-firewall.v4.12.0-202212061458

Namespace
NS openshift-ingress-node-firewall

Labels Edit
operators.coreos.com/ingress-node-firewall.openshift-ingress-node-firewall

Annotations
17 annotations

Managed Namespaces
All Namespaces

Created at
Dec 13, 2022, 10:09 AM

Status
Succeeded

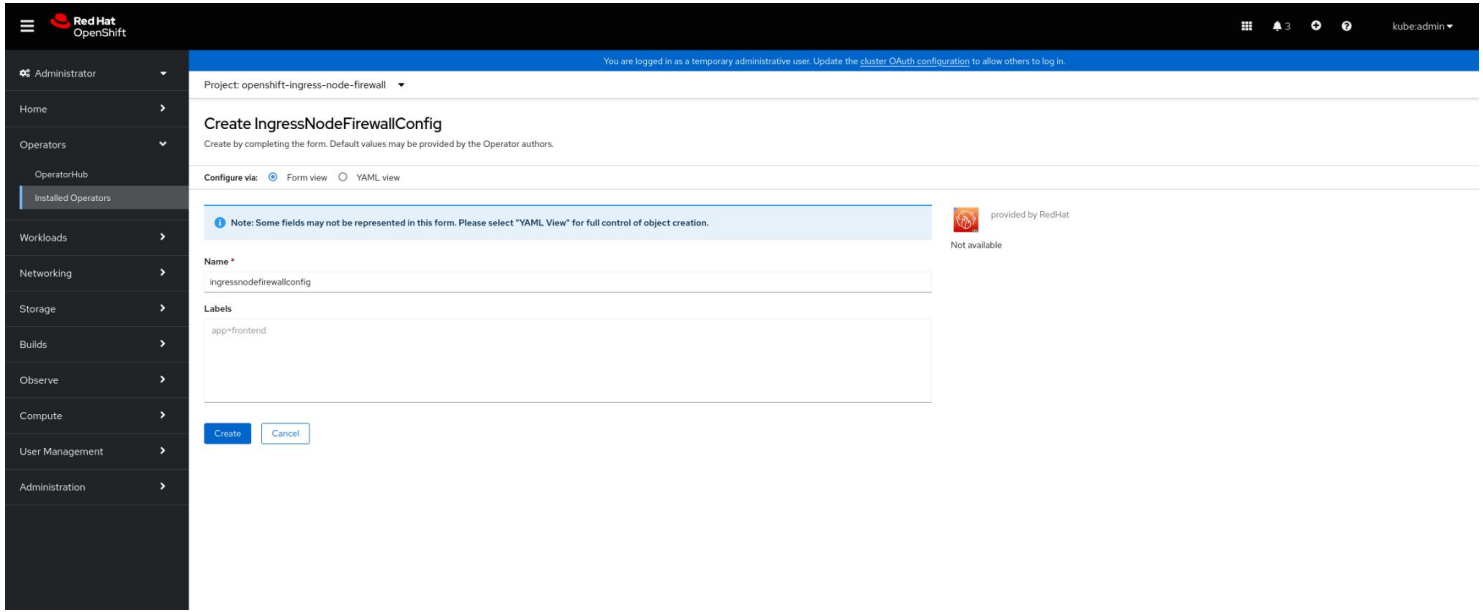
Status reason
install strategy completed with no errors

Operator Deployments
D ingress-node-firewall-controller-manager

Operator ServiceAccounts
SA ingress-node-firewall-controller-manager

OperatorGroup
OG openshift-ingress-node-firewall-j6kbp

Red Hat


Red Hat OpenShift
3
kube:admin
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Project: openshift-ingress-node-firewall
Create IngressNodeFirewallConfig
Create by completing the form. Default values may be provided by the Operator authors.
Configure via:
Form view
YAML view
Note: Some fields may not be represented in this form. Please select "YAML View" for full control of object creation.
provided by RedHat
Not available
Name *
ingressnodefirewallconfig
Labels
app=frontend
Create
Cancel

Red Hat OpenShift

kube:admin

Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: openshift-ingress-node-firewall

Remove ingress

sourceCIDRs *

sourceCIDRs defines the origin of packets that FirewallProtocolRules will be applied to.

Remove sourceCIDRs

Value *

fc00:f853:ccd:e793::0/64

Add sourceCIDRs

rules

rules is a list of per protocol ingress node firewall rules.

Remove rules

order *

10

order defines the order of execution of ingress firewall rules. The minimum order value is 1 and the values must be unique.

action

Deny

action can be Allow or Deny, default action is Allow.

protocolConfig

protocolConfig is a discriminated union of a protocol's specific configuration for TCP, UDP, SCTP, ICMP and ICMPv6. If not specified, packet matching will be based on the protocol value and protocol configuration, such as dstPort/type/code, will be ignored

protocol *

ICMPv6

protocol can be ICMP, ICMPv6, TCP, SCTP or UDP.

icmp

icmp defines an ingress node firewall rule for ICMP protocol.

icmpv6

icmpv6 defines an ingress node firewall rule for ICMPv6 protocol.

icmpCode

icmpCode defines ICMP Code ID (RFC 792). if configured, this field matches against the ICMP/ICMPv6 header otherwise its ignored.

icmpType

128

imcpType defines ICMP Type Numbers (RFC 792). if configured, this field matches against the ICMP/ICMPv6 header otherwise its ignored.

Add rules

Add ingress

interfaces

interfaces is a list of interfaces where the ingress firewall policy will be applied on.

Remove interfaces

Value *

eth0

Add interfaces

nodeSelector

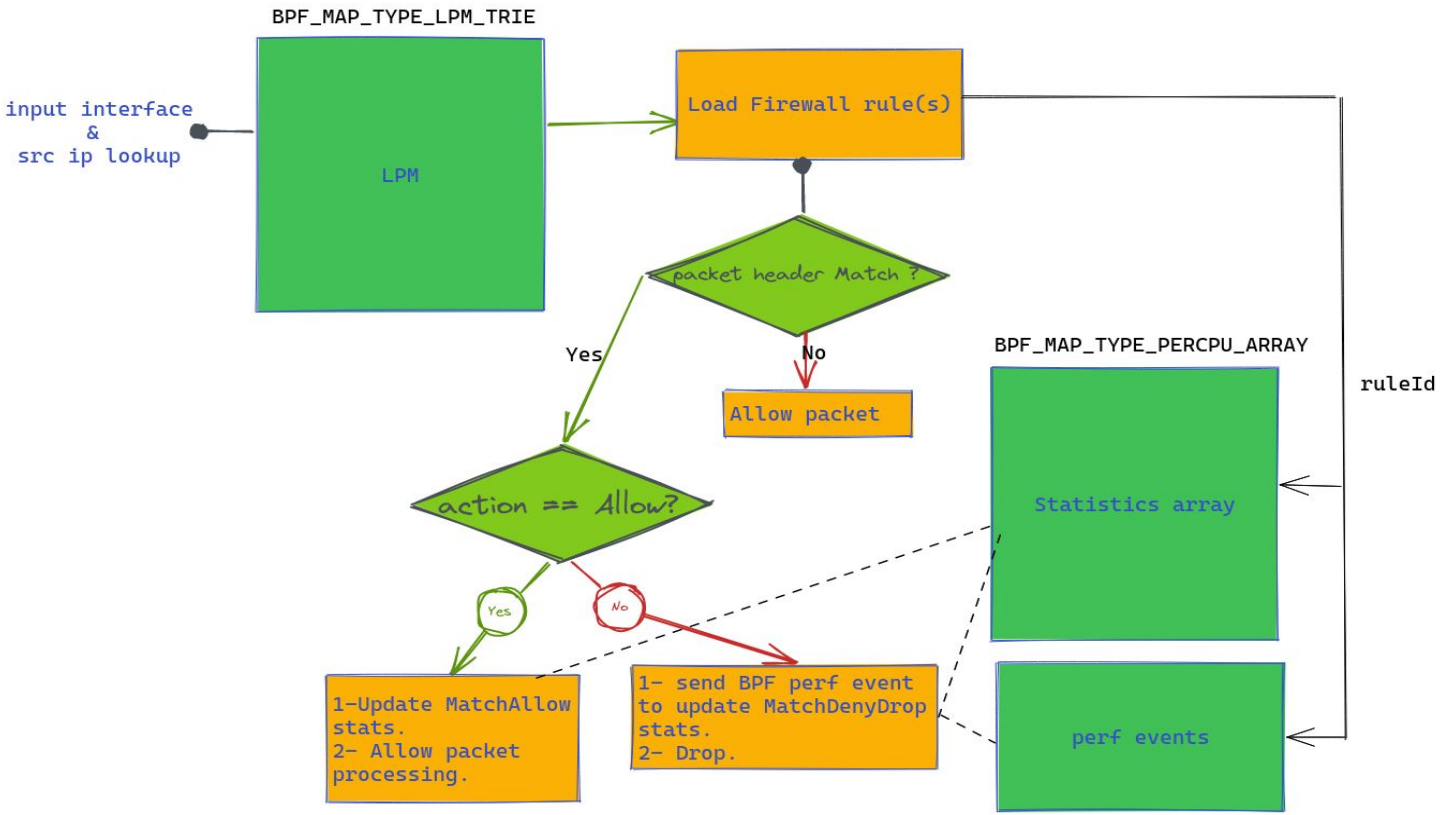
nodeSelector Selects node(s) where ingress firewall rules will be applied to.

matchExpressions

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Red Hat

Ingress node firewall XDP packet processing



Note: XDP program uses Fail open i.e if no rules match it will allow the packets.

eBPF rules statistics sample output


```
oc exec -it -n openshift-ingress-node-firewall ingress-node-firewall-daemon-pqx56  -c daemon --
bash
# HELP ingressnodefirewall_node_packet_allow_bytes The number of bytes for packets which
results in an allow IP packet result
# TYPE ingressnodefirewall_node_packet_allow_bytes gauge
ingressnodefirewall_node_packet_allow_bytes 0
# HELP ingressnodefirewall_node_packet_allow_total The number of packets which results in an
allow IP packet result
# TYPE ingressnodefirewall_node_packet_allow_total gauge
ingressnodefirewall_node_packet_allow_total 0
# HELP ingressnodefirewall_node_packet_deny_bytes The number of bytes for packets which results
in an deny IP packet result
# TYPE ingressnodefirewall_node_packet_deny_bytes gauge
ingressnodefirewall_node_packet_deny_bytes 98
# HELP ingressnodefirewall_node_packet_deny_total The number of packets which results in a deny
IP packet result
# TYPE ingressnodefirewall_node_packet_deny_total gauge
ingressnodefirewall_node_packet_deny_total 1
```

eBPF events log sample output

```
oc logs -n openshift-ingress-node-firewall ingress-node-firewall-daemon-pqx56  -c events
2022-08-25 14:50:41 +0000 UTC cnfdt19.lab.eng.tlv2.redhat.com ruleId 1 action Drop len 98 if eno1
2022-08-25 14:50:41 +0000 UTC cnfdt19.lab.eng.tlv2.redhat.com      ipv4 src addr 10.46.55.223 dst
addr 10.46.55.33
2022-08-25 14:50:41 +0000 UTC cnfdt19.lab.eng.tlv2.redhat.com      icmpv4 type 8 code 0
2022-08-25 15:01:34 +0000 UTC cnfdt19.lab.eng.tlv2.redhat.com ruleId 1 action Drop len 98 if eno1
2022-08-25 15:01:34 +0000 UTC cnfdt19.lab.eng.tlv2.redhat.com      ipv4 src addr 10.46.55.223 dst
addr 10.46.55.33
2022-08-25 15:01:34 +0000 UTC cnfdt19.lab.eng.tlv2.redhat.com      icmpv4 type 8 code 0
```

Development Testing

- Component level unit-tests for all components.
- E2E testing coverage.

Troubleshooting

- oc adm must-gather – **gather_ingress_node_firewall**
- From sos node's report it will collect eBPF bpftool outputs at */sos_commands/ebpf*
- To enable XDP lookup matching debug we add debug config to IngressNodeFirewallConfig object and redeploy

```
cat <<EOF | oc apply -f -
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewallConfig
metadata:
 name: ingressnodefirewallconfig
 namespace: openshift-ingress-node-firewall
spec:
 nodeSelector:
   node-role.kubernetes.io/worker: ""
 debug: true
EOF
```

This will create a hashmap table storing the Firewall rule lookup key and the value
Use bpftool to dump this map and match the key with the packet's header

Future releases

- Stateful Ingress node firewall to detect and block flood attacks :-
    - SYN Flood attacks
    - UDP Flood attacks
    - ICMP/ICMPv6 Flood attacks
- Large MTU support (Fixed in 4.14).
- Ability to chain multiple XDP programs to the same interface(s).
- Integrate with BPFd.

DEMO

Ingress node firewall demo on OCP cluster