CFP: Operator Manages Cilium Identities

Authors: dordel@google.com
Area: SIG: SIG-POLICY, SIG-AGENT, SIG-K8S
Sharing: **Public**
Created: 2023-08-28
Updated: 2024-10-18
Reviewers: Antonio Ojea, Joe Stringer, <add yourself>

Status: **Outdated**

Up to date information is in the CFP PR: https://github.com/cilium/design-cfps/pull/59

Github issue: https://github.com/cilium/cilium/issues/27752

Agent and Operator are short names used for cilium-agent and cilium-operator respectively.

Abbreviations for Cilium custom resources used in this document:
CID - Cilium Identity
CEP - Cilium Endpoint
CES - Cilium Endpoint Slice

Status

Reviewed and ready for implementation

There is a broad agreement for the proposed changes. There is the "Review discussion" section just below, for a few latest design details.

Review discussion

There are a few topics to discuss after the initial review. This section will be used to come to agreements for the following topics, and then the document will be updated and ready to move to the next stages -- CFP format on github and then implementation.

1) ID assignment on pod creation
Proposal: Use temporary ID (Declined)

Other options:

- init ID
- local ID
- block CNI - wait for ID (Accepted)

Benefits

- Quicker path to endpoint regeneration. Important for pods that have no labels used in peer selector of network policies, or pods that are not at all affected by network policies. Policies applied to the pod (subject selector) are going to work.
- Reduce pod startup latency.

Downsides

- In most cases, temporary ID is used on creation to regenerate the endpoint, only to regenerate again immediately, after global ID is assigned to the endpoint. However, the second regeneration will not actually need to make any changes to eBPF maps, because all entries should stay the same. Temporary ID is not used for enforcing policies applied to other pods, including pods that are running on the same node as the pod that has temporary ID. Regeneration won't be triggered for other endpoints on the node when temp ID is created, only when global ID is created, which is what happens anyway.
- [Input from Joe Stringer] "Other downsides are the complication of managing the additional identities (both the direct management, impact on policy of other endpoints, and need to prevent leaking these new identities onto the network), and potential for debugging difficulty due to the identity being more highly dependent on time and other state synchronization (basically IP to Identity binding is not as tight, during pod startup there will be blips where network traffic is associated with local identities that are ephemeral and therefore more difficult to understand - by the time you observe the flows, the identity may be removed)."
    - Temporary ID won't impact policies for other local endpoints, because it will be only used for enforcing policies that apply (subject pod selector) to the pods that have the temporary ID, and nothing else.

Other options

a) Init ID - Doesn't have the benefit of enforcing policies that apply to new pods.
b) Wait for CID - Blocking CNI and pod creation is not appropriate for network policies. Pod ready status is an option that can be added in the future, as part of the "loose" or "strict" policy enforcement.

2) Operator updates CEP's CID field after centralized global identity allocation

Proposal: Operator doesn't update identity field of CEP (Accepted)

Benefits

- CEP with an assigned global identity will be updated slightly soon. The amount of time it takes for Agents to receive the CREATE CID watch events. Estimate latency for most cases is less than 100ms.

Downsides

- Two writers to CEPs, Agents and Operator. Adds mixed ownership and potential update race to update CEP fields. It could be simply handled in the initial implementation, but maintaining and extending it will be more difficult, because it will be easier to introduce bugs.
- More complexity.

3) Operator creates CIDs from Pods, not CEPs (Accepted**)**

Benefits

- Operator can create CIDs as soon as pod objects are created.
- In most cases, CEPs will have a ready new CID even if it's a new unique label set.
- Pod startup latencies is reduced (estimated up to 100 ms on average)
- Designed to stay more flexible, relying on standard K8s resources rather than custom resources.

Downsides

- Redundant CIDs might exist for pods that are stuck in scheduling state. This is a small concern, and it's possible to mitigate with safeguards that will keep the excess of CIDs within the budget of unused identities.

Objective

Centralize CID management to a single pod (Operator), instead of distributed management, done by all Agents. The goal is to improve security, reliability, performance and scalability of CID, and enable more advanced optimizations.

Background & Motivation

CID represents complete data of one security identity, consisting of a numeric identity and a list of labels. Each CEP has an assigned identity based on its pod and namespace labels. CIDs are used to group pod and namespace labels that are used to enforce network policies.

CID duplication is a flaw of the CID management system. It occurs whenever Agents create multiple CIDs for the same set of labels. The most common case is when pods of a new deployment (new unique label set) is scheduled onto multiple nodes. Although deployment creation causes CID duplication, it doesn't happen often enough to produce many duplicate CIDs. However, namespace label changes, either on pods or namespaces, immediately trigger massive CID duplication, because all Agents react to the changes at the same time. A test with 5000 nodes shows that having 1 deployment with 5000 pods running 1 pod per node, can cause 4999 duplicate CIDs after changing the namespace labels.

CIDs are presenting issues with reliability and scalability, and are causing network policies to be in a broken state. Most notable cases:

- Reaching maximum number (65k) of CIDs in the cluster.
- eBPF policy maps overflowing with over 16k entries.
- CID garbage collection malfunction.
- Network policy incorrectly dropping connections on KCP upgrade, when cilium-agent restarts. Related to CID duplication
- Unhealthy cluster state that is difficult (or impossible) to recover from - related to numbers of IDs (and other Cilium custom resources) and CID duplication.

Design

Operator changes

Operator will start managing CIDs, and will become the only system pod that issues mutating requests for CIDs to the kube-apiserver. A new controller (CID controller) will run inside Operator that will be responsible for managing CIDs and updating CEPs.

1) Operator requires watching the following resources to allocate CIDs and assign them to CEPs:

- CEPs / Pods (Already watches) - To manage CIDs based on pod labels and update CEPs with correct CIDs.
- Namespaces - to create and delete CIDs based on namespace labels changes.
- CIDs - to keep the desired state of CIDs. Required for:
  - Getting the initial state on start-up.
  - Being compatible with Agents managing CIDs simultaneously.
  - Handling unplanned CID changes (create, update, delete) by other sources -- not Operator or Agents.

2) Operator will modify the following resources:

- CIDs - Create and delete. CIDs shouldn't be modified. If namespace or pod labels are changed, new CIDs will be created for all unique label sets that don't match any existing identity. Unused CIDs will be cleaned up by the CID GC.
- CEPs - Update the `Status.Identity` field.

3) CID GC reworked:

- Heartbeat status annotations will no longer be used. It will not be added to CIDs by the Operator, or removed by the Agent. Operator will remain to be compatible with Agents managing CIDs simultaneously. Agents just won't ever have a chance to remove the annotation.
- CIDs will still be GC-ed periodically -- every 10 minutes.

- CIDs will be deleted only when they aren't used anywhere, in CID, CEP and CES (if enabled) controller's caches and watcher stores.

4) On startup, the CID controller will cache existing CIDs from the CID watcher store and usage of CIDs in CEPs from the CEP watcher store. The cache will be used for mapping CID name (256 - 65k integer) to security labels and the reverse mapping -- labels to CID name.

5) The CID controller will periodically run CID reconciliation for all CEPs, alongside CID GC, on startup and every 10 minutes. It will make sure that all CEPs have correct CIDs assigned to them.

6) There will be no changes to the CES controller running inside Operator.

a) The CES controller receives CEP updates with assigned CIDs, after the CID controller updates them, and then batch them into CESs.
b) The CES controller ignores CEPs that have empty (nil) `Status.Identity`. This means that for new CEPs, the CES controller skips CEP creation events that have empty CID, and then batches them on the next immediate update events from the CID controller, where CIDs are assigned.

Agent changes

Agents will stop managing global security identities (CIDs). Agents will no longer perform following actions:

- Allocate security identities in the global range [256, 65536].
- Create CIDs. Agents will no longer need the permission to modify CIDs.
- Remove heartbeat status annotation on CIDs (part of CID GC).

Temporary ID
A new type of local ID will be used, called temporary ID. It will be used only locally by each node for the endpoints that they own. It's needed to allow endpoint regeneration before a global (cluster-wide) identity is assigned to CEPs.

Agents will still watch CIDs. When a new pod is created or labels are changed, it will lookup the CID store to match the labels to a CID. Temporary ID will only be created when no matching CID is found.

For new CEPs, temporary ID is allocated and assigned locally in the Endpoint Manager's cache, before it's overwritten by the global CID that Operator creates. Temporary ID will not be assigned to CEP's `Status.Identity` field, because it's only relevant locally to the node that owns the CEP. A new range of 1024 (2^10) is reserved from the first immediate available range that is reserved for future use. 1024 is more than safe to have per node, as there should not be that many pods on one node at all. (256 is the maximum number of pods per node on GKE).

Security identity distribution:

```
0x00000001 - 0x000000FF (1         to 2^8  - 1)            => reserved identities
0x00000100 - 0x0000FFFF (2^8        to 2^16 - 1)           => cluster-local identities
0x00010000 - 0x00FFFFFF (2^16       to 2^24 - 1)           => identities for remote clusters
0x01000000 - 0x0100FFFF (2^24       to 2^24 + 2^16 - 1)    => identities for CIDRs (node-local)
0x01010000 - 0xFFFFFFFF (2^24 + 2^16 to 2^32 - 1)          => reserved for future use
```

Add a new reserved range of size 1024 from 2^24 + 2^16 to 2^24 + 2^16 + 2^10.

```
0x01010000 - 0x010103FF (2^24 + 2^16 to 2^24 + 2^16 + 2^10 - 1)        => temporary identities for local endpoints
```

Namespace watch
Agents will continue to watch namespaces in order to instantly update security identity labels in the local Endpoint manager's cache. More info below in Label changes: Namespace.

Pod startup
CNI triggers CEP creation for every new pod. Pod can be marked ready, only after CEP is created and endpoint regenerated. CEP creation used to require a global ID to be allocated and assigned to CEP. Now, a temporary ID will be assigned immediately, and later overwritten by the global ID received through CESs.

Pod startup latency will be reduced when a CID for pod's labels doesn't exist yet. CID creation is time consuming mainly because of waiting for kube-apiserver to create the CID resource in Etcd.

Policy enforcement
Functionality-wise for the host node, the changes have no downsides for network policy enforcement. New endpoints can be regenerated even without a global ID assigned. Network policies will be enforced based on already known CIDs from remote pods.

There might be a very short additional latency for remote notes to receive the CEP with updated CID, because Operator needs to receive the CEP without CID assigned and then update it with a newly allocated CID.
Old process:
CID created (Agent) -> CEP created (Agent) -> Agents get updated CEP

New process:
CEP created (Agent) -> Operator gets CEP -> CID created (Operator) -> CEP updated (Operator) -> Agents get updated CEP

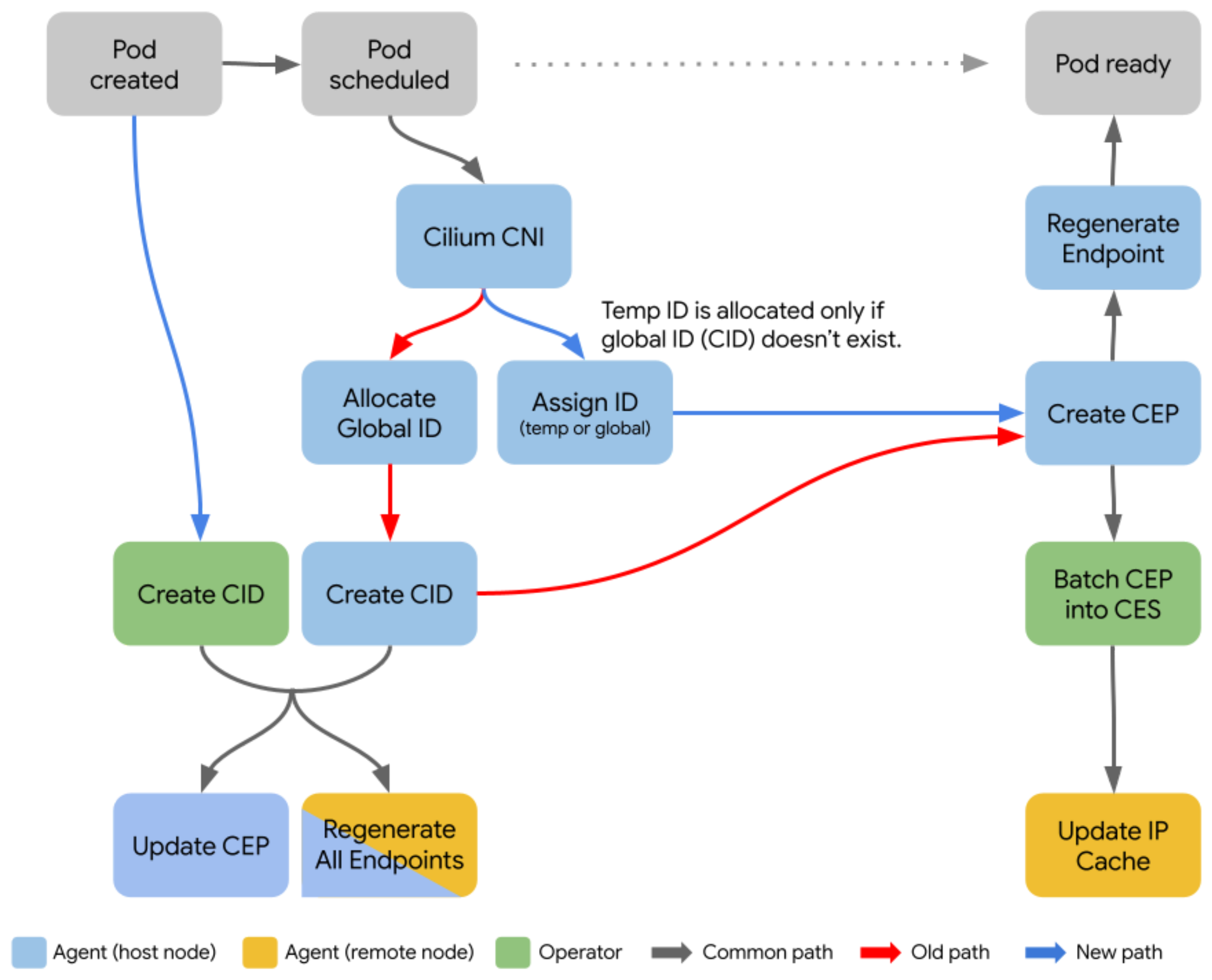
Pod creation process

CID assignment paths on pod creation:

1. Regular path: Pod created -> CID created (Operator) -> Pod scheduled -> Cilium CNI -> CID assigned

2. Missing global ID path: Pod created -> Pod scheduled -> Cilium CNI -> Temp CID assigned

The diagram below covers high level actions that happen when a pod is created. From pod scheduled to pod ready, and to CEP and CID data being received by all nodes in the cluster. CEP batching into CES is included, but that's an optional step based on configuration. Comparison before (red arrows) and after (blue arrows) changes.



CEP creation in parallel triggers its endpoint regeneration on the host node and CEP propagation to all other nodes.

Endpoint regeneration happens when:

- An endpoint is created and updated, including pod and namespace label changes (security identity change).
- CID is created or deleted - all Agents regenerate all of their endpoints.
- Network policies are changed.

IPCache is updated when remote CEPs are received.

Pod ready status

Pod is ready to start participating in the network when the pod status is set to ready (Pod network readiness). Service programming happens only after endpoints are network ready. This is important because new pods can be marked as ready before other nodes know about their IP->CID mappings. It can negatively impact egress network policies, because service traffic will be forwarded to the new pod, and network policy will deny the connection, since there is no information about the identity of the destination IP yet.

Options:

1) Proposed option - Pod becomes ready immediately after CEP creation and endpoint regeneration. It can still have a temporary ID assigned, while waiting for a global security identity (CID) to be assigned and propagated to other nodes.

   Proposal: To be part of the implementation
   Reason: No changes. It can already happen that a pod behind a service is having network policies to incorrectly deny traffic when there is high propagation latency.

   The risk of this issue causing disruption is extremely low. Performance of Cilium resource propagation is very high, and only in very rare cases can it happen that it grows more than a few seconds. It's measured by CEP propagation delay, and tracked to be an SLI for network policies.

2) Pod becomes ready only after the CEP with CID assigned is received back at the host. This is only for pods affected by network policies that don't have an existing matching CID in the CID store. The rest of the pods are ready immediately.

   Proposal: Potential future adjustment
   Reason: Not related to or required for Operator managing CIDs.
   It has existed in DPv2/Cilium for a long time and should be tracked as a separate improvement to network policy enforcement, which is out of the scope of this design document.

Two CEP writing sources

When Operator starts managing CIDs, there will be two sources that will be making changes to CEPs, Operator and Agent. This means that Operator and Agent should work in harmony, not racing or overriding each other's changes.

Agent
Create CEP - On pod creation
Update CEP - On pod update. Managing any changes to the CEP, such as Status and labels.

Delete CEP - On pod deletion

Agents will assign CEP's `Status.Identity` field only when the matching CID is found in the CID watcher cache for new CEPs or when pod labels are changed. On CEP creation, a nil value is assigned, while on CEP updates, the old identity will remain on the CEP until Operator reassigns it.

Operator
Update CEP - On CEP or namespace label changes. Updates only the `Status.Identity`.
There will be no collision with Agent updates, because Agent will only update the identity of CEP when it finds a matching CID in the CID watcher cache. In this case, Operator will validate it and not update it again. When pod labels change, Agent will update CEP labels, and only then Operator receive the CEP with changed labels, and update the CID. Operator will also reconcile all CEPs every specified interval (e.g. 10 minutes) regardless of any CEP or namespace events.

Label changes

Pod

When deployment labels change, all Agents hosting the pods will update labels and identities for their CEPs for the changed pods. Agents will locally try to match the new label set to an existing CID. If a match is not found, Agents will create a temporary ID, until it's replaced by the CID created by the Operator.

Agents will assign CEP's `Status.Identity` field only when the matching CID is found in the CID watcher cache for new CEPs or when pod labels are changed. On CEP creation, a nil value is assigned, while on CEP updates, the old identity will remain on the CEP until Operator reassigns it.

Namespace

Note: CEP object contains namespace labels as part of the security identity labels only within the `Status.Identity` field.

When namespace labels change, the Operator will run identity reconciliation for all CEPs in the changed namespace, which will assign correct identities to CEPs (with new CIDs created). Old CIDs will no longer be used and will be deleted by CID GC.

Agents will continue to watch namespaces only to update local Endpoint manager's cache with the correct namespace labels. Agents processing namespace label changes won't trigger creation or changes of CIDs, nor updates to CEPs, because Operator will be responsible for this. This works differently from pod label changes, because Operator is acting on CEP and namespace changes, while Agents are acting on pod changes. Agents will locally assign temporary IDs to all CEPs in the namespace that has labels changed, but not change the identity on CEP objects. Temporary IDs are useful here because all endpoints will be able to be regenerated immediately, without waiting for CIDs to be created, assigned and propagated. Regeneration of endpoints means that correct policies will be enforced for the new labels. Agents will receive CEPs with updated CID that matches the new namespace labels.

Simultaneous CID management compatibility

It’s important for Operator and Agents to be compatible when both are managing CIDs at the same time, because upgrading KCP depends on it. This is achieved by moving the CID management functionality to Operator to precisely do the work to get the same results as when Agents are managing CIDs. As long as Agents are managing CIDs, the Operator will only be verifying that the CID and CEP states are correct, without making any modifications. The new CID GC requires no changes in Agents.

The only problem is namespace label changes, because both Operator and Agents will be watching namespaces to react on namespace label changes and create new CIDs and assign them to CEPs as soon as possible. Namespace label change will initially generate duplicate CIDs created from Agents (happens anyway) and Operator (additionally). Operator will receive CEP updates made by Agents and will accept them. Operator will later clean up unused CIDs.

CID deduplication

Operator will not create duplicate CIDs, but it will allow for their existence. In case other sources, like Agents, are making duplicate CIDs, Operator will be compatible with tracking multiple CIDs. This is needed to make Operator compatible with Agents managing CIDs at the same time.

After migration to Operator managing CIDs is completed, deduplication logic should be added as an additional measure of safety.

KVStore compatibility

KVStore is a key-value store that is mostly used for sharing Cilium mappings between IPs, identities (IPCache) and labels, across the cluster (what's inside Cilium’s KVStore).

Operator managing CIDs is intended to work with `identity-allocation-mode=CRD`, which is the default.

All features that rely on using kvstore will not work until there is an equivalent kvstore implementation of Operator managing CIDs.

- Cluster-mesh - KVStore is required for connecting multiple clusters, because the information about pod IPs and CIDs is shared with KVStore watchers.

Enablement

The feature will be guarded by a flag `operator-manage-cid`, that can be set in container arguments in Operator's and Agent's manifests. The default values for both will be `false`, meaning that Operator doesn't manage CIDs, while Agents do.

Release Operator first

Releasing Operator first will allow us to first ensure that Operator code is compatible with Agents simultaneously managing CIDs. Then we can have higher confidence that upgrading to Agent's not managing CIDs is safe.

Long term plan

The long term plan is to make Operator managing CIDs the default, and deprecate the part of Agent that is currently used to manage CIDs.

Migration

Any situation when neither Operator nor Agents are managing CIDs is effectively a downtime for network policy changes. The control plane side of network policies and other features relying on CIDs will not carry out any related changes during this time, causing them to stop working.

Upgrade

Safe upgrade, without downtime, requires simultaneous CID management compatibility.

If we upgrade the Operator from a version that is not managing CIDs to a version that is managing CIDs while Agents are also managing CIDs, there should be no negative impact on the system. Everything should work the same.

This works well with the way KCP upgrade works. Operator is upgraded first when KCP nodes are restarted with the updated manifests. Agent daemonset updates with a rolling update, only after all KCP nodes are running and healthy with the new version of Operator. While Operator is already managing CIDs, Agent pods are restarted to a new version, and they stop managing CIDs. The new Agents will create CEPs with an empty (nil) assigned identity when there is no matching CID in watcher cache for a new pod. Operator will then take care of those CEPs, by creating and assigning new CIDs, on CEP watcher events and periodically on full reconciliation rounds.

Downgrade

Downgrade without downtime is possible only if Agent daemonset is downgraded first -- updated to an earlier version where Agent manages CIDs. Otherwise, if KCP is downgraded first, there will be a downtime lasting at least the amount of time needed for Agent daemonset to be downgraded after the KCP downgrade.

Security

- Agents will no longer need to have the RBAC permission for modifying CIDs.

Future enhancements

CID lazy creation

Scalability and performance optimization

Create IDs only for labels used in network policies to greatly reduce the number of CIDs. Only pod labels used in the peer pod label selector of network policies will be relevant for ID creation.

There are multiple levels of optimization and complexity this solution can go to. The proposal is to start with a basic implementation that will have effects like ID relevant labels, based on label keys from network policy selectors. Discussion of more advanced solutions is outside the scope of this document.

Operator acts on Pod instead of CEP changes

Performance optimization

Operator doesn't watch for CEPs but watches for Pods instead. As soon as a pod is created (not started) Operator allocates a CID immediately. This way the new CID is getting created in parallel as the pod is getting scheduled.

Downside: In case pods are unschedulable we will be allocating identities even though they might not be used at the end.

This topic entails the possibility of removing CEPs altogether, because we can just have CIDs and CESs. Agents don’t create CEPs anymore, just watch for CESs and CIDs. Operator creates both CIDs and CESs with all info gathered from pods.

Loose and Strict modes for pod readiness

An option to make pods ready only when network policies are enforced. It is possible to use the pod readiness gate to control when pods are marked as ready.

With **Loose** mode, the default, pods will be ready immediately after endpoint regeneration, without waiting for policy enforcement -- CEPs (or CESs) to be propagated to remote nodes.

With **Strict** mode, all pods affected by network policies will become ready only after network policies are enforced on remote nodes. The indicator for that will be when CEPs (or CESs) are propagated back to the host nodes. This will allow users to have network policy enforcement consistency for new pods -- no denied traffic because policy enforcement-relevant data is not propagated yet.