Security

Astra Trident

NetApp
October 31, 2024

Table of Contents

Security

Security

Use the recommendations listed here to ensure your Astra Trident installation is secure.

Run Astra Trident in its own namespace

It is important to prevent applications, application administrators, users, and management applications from accessing Astra Trident object definitions or the pods to ensure reliable storage and block potential malicious activity.

To separate the other applications and users from Astra Trident, always install Astra Trident in its own Kubernetes namespace (`trident`). Putting Astra Trident in its own namespace assures that only the Kubernetes administrative personnel have access to the Astra Trident pod and the artifacts (such as backend and CHAP secrets if applicable) stored in the namespaced CRD objects.
You should ensure that you allow only administrators access to the Astra Trident namespace and thus access to the `tridentctl` application.

Use CHAP authentication with ONTAP SAN backends

Astra Trident supports CHAP-based authentication for ONTAP SAN workloads (using the `ontap-san` and `ontap-san-economy` drivers). NetApp recommends using bidirectional CHAP with Astra Trident for authentication between a host and the storage backend.

For ONTAP backends that use the SAN storage drivers, Astra Trident can set up bidirectional CHAP and manage CHAP usernames and secrets through `tridentctl`.
Refer to to understand how Astra Trident configures CHAP on ONTAP backends.

Use CHAP authentication with NetApp HCI and SolidFire backends

NetApp recommends deploying bidirectional CHAP to ensure authentication between a host and the NetApp HCI and SolidFire backends. Astra Trident uses a secret object that includes two CHAP passwords per tenant. When Astra Trident is installed, it manages the CHAP secrets and stores them in a `tridentvolume` CR object for the respective PV. When you create a PV, Astra Trident uses the CHAP secrets to initiate an iSCSI session and communicate with the NetApp HCI and SolidFire system over CHAP.

> The volumes that are created by Astra Trident are not associated with any Volume Access Group.

Use Astra Trident with NVE and NAE

NetApp ONTAP provides data-at-rest encryption to protect sensitive data in the event a disk is stolen, returned, or repurposed. For details, refer to Configure NetApp Volume Encryption overview.

- If NAE is enabled on the backend, any volume provisioned in Astra Trident will be NAE-enabled.
- If NAE is not enabled on the backend, any volume provisioned in Astra Trident will be NVE-enabled unless you set the NVE encryption flag to `false` in the backend configuration.

> Volumes created in Astra Trident on an NAE-enabled backend must be NVE or NAE encrypted.
>
> - You can set the NVE encryption flag to `true` in the Trident backend configuration to override the NAE encryption and use a specific encryption key on a per volume basis.
> - Setting the NVE encryption flag to `false` on an NAE-enabled backend will create an NAE-enabled volume. You cannot disable NAE encryption by setting the NVE encryption flag to `false`.

- You can manually create an NVE volume in Astra Trident by explicitly setting the NVE encryption flag to `true`.

For more information on backend configuration options, refer to:

- ONTAP SAN configuration options
- ONTAP NAS configuration options

Linux Unified Key Setup (LUKS)

You can enable Linux Unified Key Setup (LUKS) to encrypt ONTAP SAN and ONTAP SAN ECONOMY volumes on Astra Trident. Astra Trident supports passphrase rotation and volume expansion for LUKS-encrypted volumes.

In Astra Trident, LUKS-encrypted volumes use the aes-xts-plain64 cypher and mode, as recommended by NIST.

Before you begin

- Worker nodes must have cryptsetup 2.1 or higher (but lower than 3.0) installed. For more information, visit Gitlab: cryptsetup.
- For performance reasons, we recommend that worker nodes support Advanced Encryption Standard New Instructions (AES-NI). To verify AES-NI support, run the following command:

  ```
  grep "aes" /proc/cpuinfo
  ```

  If nothing is returned, your processor does not support AES-NI. For more information on AES-NI, visit: Intel: Advanced Encryption Standard Instructions (AES-NI).

Enable LUKS encryption

You can enable per-volume, host-side encryption using Linux Unified Key Setup (LUKS) for ONTAP SAN and ONTAP SAN ECONOMY volumes.

Steps

1. Define LUKS encryption attributes in the backend configuration. For more information on backend configuration options for ONTAP SAN, refer to ONTAP SAN configuration options.

```
"storage": [
    {
        "labels":{"luks": "true"},
        "zone":"us_east_1a",
        "defaults": {
            "luksEncryption": "true"
        }
    },
    {
        "labels":{"luks": "false"},
        "zone":"us_east_1a",
        "defaults": {
            "luksEncryption": "false"
        }
    },
]
```

2. Use `parameters.selector` to define the storage pools using LUKS encryption. For example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. Create a secret that contains the LUKS passphrase. For example:

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

Limitations

LUKS-encrypted volumes cannot take advantage of ONTAP deduplication and compression.

Backend configuration for importing LUKS volumes

To import a LUKS volume, you must set `luksEncryption` to `true` on the backend. The `luksEncryption` option tells Astra Trident if the volume is LUKS-compliant (`true`) or not LUKS-compliant (`false`) as shown in the following example.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

Rotate a LUKS passphrase

You can rotate the LUKS passphrase and confirm rotation.

> Do not forget a passphrase until you have verified it is no longer referenced by any volume, snapshot, or secret. If a referenced passphrase is lost, you might be unable to mount the volume and the data will remain encrypted and inaccessible.

About this task

LUKS passphrase rotation occurs when a pod that mounts the volume is created after a new LUKS passphrase is specified. When a new pod is created, Astra Trident compares the LUKS passphrase on the volume to the active passphrase in the secret.

- If the passphrase on the volume does not match the active passphrase in the secret, rotation occurs.
- If the passphrase on the volume matches the active passphrase in the secret, the `previous-luks-passphrase` parameter is ignored.

Steps

1. Add the `node-publish-secret-name` and `node-publish-secret-namespace` StorageClass parameters. For example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. Identify existing passphrases on the volume or snapshot.

   Volume

   ```
   tridentctl -d get volume luks-pvc1
   GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

   ...luksPassphraseNames:["A"]
   ```

   Snapshot

   ```
   tridentctl -d get snapshot luks-pvc1
   GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

   ...luksPassphraseNames:["A"]
   ```

3. Update the LUKS secret for the volume to specify the new and previous passphrases. Ensure `previous-luke-passphrase-name` and `previous-luks-passphrase` match the previous passphrase.

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: luks-pvc1
   stringData:
     luks-passphrase-name: B
     luks-passphrase: secretB
     previous-luks-passphrase-name: A
     previous-luks-passphrase: secretA
   ```

4. Create a new pod mounting the volume. This is required to initiate the rotation.
5. Verify the the passphrase was rotated.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

Results

The passphrase was rotated when only the new passphrase is returned on the volume and snapshot.

> ⓘ If two passphrases are returned, for example `luksPassphraseNames: ["B", "A"]`, the rotation is incomplete. You can trigger a new pod to attempt to complete the rotation.

Enable volume expansion

You can enable volume expansion on a LUKS-encrypted volume.

Steps

1. Enable the `CSINodeExpandSecret` feature gate (beta 1.25+). Refer to Kubernetes 1.25: Use Secrets for Node-Driven Expansion of CSI Volumes for details.
2. Add the `node-expand-secret-name` and `node-expand-secret-namespace` StorageClass parameters. For example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

Results

When you initiate online storage expansion, the kubelet passes the appropriate credentials to the driver.