



## **Security**

### **Astra Trident**

NetApp  
March 28, 2023

# Table of Contents

- Security ..... 1
  - Security ..... 1
  - Linux Unified Key Setup (LUKS) ..... 2

# 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`.

See [here](#) to understand how Astra Trident configures CHAP on ONTAP backends.



CHAP support for ONTAP backends is available with Trident 20.04 and later.

### 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 Trident is installed as a CSI provisioner, it manages the CHAP secrets and stores them in a `tridentvolume` CR object for the respective PV. When you create a PV, CSI 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 CSI Trident are not associated with any Volume Access Group.

In the non-CSI frontend, the attachment of volumes as devices on the worker nodes is handled by Kubernetes. After volume creation, Astra Trident makes an API call to the NetApp HCI/SolidFire system to retrieve the secrets if the secret for that tenant does not already exist. Astra Trident then passes the secrets on to Kubernetes. The kubelet located on each node accesses the secrets via the Kubernetes API and uses them to run/enable CHAP between each node accessing the volume and the NetApp HCI/SolidFire system where the volumes are located.

## 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: netapp.io/trident  
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.

## 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-luks-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: netapp.io/trident
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.



## Copyright information

Copyright © 2023 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

## Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.