보안

Astra Trident

NetApp
April 04, 2024

목차

보안

보안

여기에 나열된 권장 사항을 사용하여 Astra Trident 설치가 안전한지 확인합니다.

자체 네임스페이스에서 **Astra Trident**를 실행합니다

애플리케이션, 애플리케이션 관리자, 사용자 및 관리 애플리케이션에서 Astra Trident 객체 정의 또는 Pod에 액세스하여 안정적인 스토리지를 보장하고 잠재적인 악성 활동을 차단하는 것이 중요합니다.

다른 애플리케이션과 사용자를 Astra Trident에서 분리하려면 항상 고유한 Kubernetes 네임스페이스에 Astra Trident를 설치하십시오 (`trident`)를 클릭합니다. Astra Trident를 자체 네임스페이스에 두면 Kubernetes 관리 담당자만 Astra Trident POD와 이름이 같은 CRD 객체에 저장된 아티팩트(예: 백엔드 및 CHAP 암호)에 액세스할 수 있습니다.
관리자만이 Astra Trident 네임스페이스에 액세스하고 에 액세스할 수 있도록 허용해야 합니다 `tridentctl` 응용 프로그램.

ONTAP SAN 백엔드에 **CHAP** 인증을 사용합니다

Astra Trident는 ONTAP SAN 워크로드에 대한 CHAP 기반 인증을 지원합니다(사용 `ontap-san` 및 `ontap-san-economy` 드라이버). 호스트와 스토리지 백엔드 간의 인증을 위해 Astra Trident와 양방향 CHAP를 사용하는 것이 좋습니다.

SAN 스토리지 드라이버를 사용하는 ONTAP 백엔드의 경우 Astra Trident는 양방향 CHAP를 설정하고 를 통해 CHAP 사용자 이름 및 암호를 관리할 수 있습니다 `tridentctl`.
을 참조하십시오 "여기" Astra Trident가 ONTAP 백엔드에서 CHAP를 구성하는 방법을 이해합니다.

ONTAP 백엔드에 대한 CHAP 지원은 Trident 20.04 이상에서 사용할 수 있습니다.

NetApp HCI 및 **SolidFire** 백엔드에서 **CHAP** 인증을 사용합니다

양방향 CHAP를 구축하여 호스트와 NetApp HCI 및 SolidFire 백엔드 간의 인증을 보장하는 것이 좋습니다. Astra Trident는 테넌트당 2개의 CHAP 암호를 포함하는 비밀 객체를 사용합니다. Trident를 CSI 프로비저닝자로 설치하면 CHAP 암호를 관리하고 에 저장합니다 `tridentvolume` 해당 PV에 대한 CR 개체입니다. PV를 생성할 때 CSI Astra Trident는 CHAP 암호를 사용하여 iSCSI 세션을 시작하고 CHAP를 통해 NetApp HCI 및 SolidFire 시스템과 통신합니다.

CSI Trident에서 생성한 볼륨은 볼륨 액세스 그룹과 연결되지 않습니다.

CSI가 아닌 프런트엔드에서는 작업자 노드의 디바이스로 볼륨을 연결하는 작업을 Kubernetes에서 처리합니다. 볼륨 생성 후 Astra Trident는 NetApp HCI/SolidFire 시스템에 API 호출을 통해 해당 테넌트의 암호가 아직 없는 경우 비밀을 검색합니다. 그런 다음 Astra Trident가 Kubernetes에 비밀을 전달합니다. 각 노드에 위치한 kubelet은 Kubernetes API를 통해 기밀에 액세스하고 이를 사용하여 볼륨에 액세스하는 각 노드와 볼륨이 있는 NetApp HCI/SolidFire 시스템 간에 CHAP를 실행/사용하도록 설정합니다.

NVE와 NAE가 포함된 Astra Trident를 사용하십시오

NetApp ONTAP는 유휴 데이터 암호화를 제공하여 디스크를 도난, 반환 또는 용도 변경할 때 중요한 데이터를 보호합니다. 자세한 내용은 을 참조하십시오 "NetApp 볼륨 암호화 구성 개요".

- 백엔드에서 NAE가 활성화된 경우 Astra Trident에 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다.
- NAE가 백엔드에서 활성화되지 않은 경우, NVE 암호화 플래그를 로 설정하지 않으면 Astra Trident에 프로비저닝된 모든 볼륨은 NVE를 사용할 수 있습니다 `false` 백엔드 구성

> NAE 지원 백엔드의 Astra Trident에 생성된 볼륨은 NVE 또는 NAE 암호화여야 합니다.
>
> - NVE 암호화 플래그를 로 설정할 수 있습니다 `true` Trident 백엔드 구성에서 NAE 암호화를 재정의하고 볼륨별로 특정 암호화 키를 사용합니다.
> - NVE 암호화 플래그를 로 설정합니다 `false` NAE 지원 백엔드에서 NAE 지원 볼륨을 생성합니다. NVE 암호화 플래그를 로 설정하여 NAE 암호화를 비활성화할 수 없습니다 `false`.

- NVE 암호화 플래그를 명시적으로 로 설정하여 Astra Trident에서 NVE 볼륨을 수동으로 생성할 수 있습니다 `true`.

백엔드 구성 옵션에 대한 자세한 내용은 다음을 참조하십시오.

- "ONTAP SAN 구성 옵션"
- "ONTAP NAS 구성 옵션"

Linux 통합 키 설정(LUKS)

LUKS(Linux 통합 키 설정)를 활성화하여 Astra Trident에서 ONTAP SAN 및 ONTAP SAN 경제 볼륨을 암호화할 수 있습니다. Astra Trident는 LUKS 암호화 볼륨에 대한 암호 순환 및 볼륨 확장을 지원합니다.

Astra Trident에서 LUKS 암호화 볼륨은 에서 권장하는 대로 AES-XTS-ai64 cypher 및 모드를 사용합니다 "NIST".

시작하기 전에

- 작업자 노드에는 Cryptsetup 2.1 이상(3.0 이하)이 설치되어 있어야 합니다. 자세한 내용은 를 참조하십시오 "Gitlab: cryptsetup".
- 성능상의 이유로 작업자 노드는 AES-NI(Advanced Encryption Standard New Instructions)를 지원하는 것이 좋습니다. AES-NI 지원을 확인하려면 다음 명령을 실행합니다.

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

  아무 것도 반환되지 않으면 프로세서는 AES-NI를 지원하지 않습니다. AES-NI에 대한 자세한 내용은 다음 웹 사이트를 참조하십시오. "인텔: AES-NI(Advanced Encryption Standard Instructions)".

LUKS 암호화를 사용합니다

ONTAP SAN 및 ONTAP SAN 이코노미 볼륨에 대해 Linux 통합 키 설정(LUKS)을 사용하여 볼륨별 호스트 측 암호화를 활성화할 수 있습니다.

단계

1. 백엔드 구성에서 LUKS 암호화 속성을 정의합니다. ONTAP SAN의 백엔드 구성 옵션에 대한 자세한 내용은 을 참조하십시오 "ONTAP SAN 구성 옵션".

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

2. 사용 `parameters.selector` LUKS 암호화를 사용하여 스토리지 풀을 정의합니다. 예를 들면 다음과 같습니다.

```
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. LUKS 암호를 포함하는 암호를 생성합니다. 예를 들면 다음과 같습니다.

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

제한 사항

LUKS - 암호화된 볼륨은 ONTAP 중복 제거 및 압축을 활용할 수 없습니다.

LUKS 볼륨을 가져오기 위한 백엔드 구성입니다

LUKS 볼륨을 가져오려면 을 설정해야 합니다 `luksEncryption` 를 선택합니다(`true` 백엔드에서. 를 클릭합니다 `luksEncryption` 옵션은 볼륨이 LUKS를 준수하는지 Astra Trident에 알려줍니다 (`true`) 또는 LUKS를 준수하지 않습니다 (`false`)를 참조하십시오.

```
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'
```

LUKS 암호를 회전합니다

LUKS 암호를 회전하고 회전을 확인할 수 있습니다.

> 볼륨, 스냅샷 또는 비밀이 더 이상 참조하지 않음을 확인할 때까지 암호문을 잊지 마십시오. 참조된 암호가 손실된 경우 볼륨을 마운트할 수 없으며 데이터가 암호화된 상태로 유지되고 액세스할 수 없게 됩니다.

이 작업에 대해

LUKS 암호 회전은 새 LUKS 암호를 지정한 후 볼륨을 마운트하는 POD가 생성될 때 발생합니다. 새 POD를 생성할 때 Astra Trident는 볼륨의 LUKS 암호를 비밀의 활성 패스프레이즈(passphrase)와 비교합니다.

- 볼륨의 암호가 비밀의 활성 암호와 일치하지 않으면 회전이 발생합니다.
- 볼륨의 암호가 비밀의 활성 암호와 일치하면 가 됩니다 `previous-luks-passphrase` 매개 변수는 무시됩니다.

단계

1. 를 추가합니다 `node-publish-secret-name` 및 `node-publish-secret-namespace` StorageClass 매개 변수입니다. 예를 들면 다음과 같습니다.

```
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. 볼륨 또는 스냅숏에서 기존 암호를 식별합니다.

   볼륨

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

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

   스냅샷

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

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

3. 볼륨에 대한 LUKS 암호를 업데이트하여 새 암호 및 이전 암호 문구를 지정합니다. 확인합니다 `previous-luke-passphrase-name` 및 `previous-luks-passphrase` 이전 패스프레이즈를 일치시킵니다.

```
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. 볼륨을 마운트하는 새 포드를 생성합니다. 이 작업은 회전을 시작하는 데 필요합니다.
5. 패스프레이즈가 회전되었는지 확인합니다.

볼륨

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

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

스냅샷

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

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

결과

볼륨과 스냅샷에 새 암호문만 반환되면 암호가 회전되었습니다.

예를 들어, 두 개의 암호 구문이 반환되는 경우 `luksPassphraseNames: ["B", "A"]`, 회전이 완료되지 않았습니다. 새 포드를 트리거하여 회전을 완료할 수 있습니다.

볼륨 확장을 설정합니다

LUKS 암호화 볼륨에서 볼륨 확장을 활성화할 수 있습니다.

단계

1. 를 활성화합니다 `CSINodeExpandSecret` 기능 게이트(베타 1.25+). 을 참조하십시오 "Kubernetes 1.25: CSI 볼륨의 노드 기반 확장에 비밀을 사용합니다" 를 참조하십시오.
2. 를 추가합니다 `node-expand-secret-name` 및 `node-expand-secret-namespace` StorageClass 매개 변수입니다. 예를 들면 다음과 같습니다.

```
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
```

결과

온라인 저장소 확장을 시작할 때 kubelet은 적절한 자격 증명을 드라이버에 전달합니다.

저작권 정보

상표 정보