trutz.io

SPICKZETTEL · VAULT → OPENBAO

Die Vault-Exit-Inventur

Zeitleiste, Inventur-Bogen, die drei Töpfe und die Cutover-Reihenfolge. Begleitmaterial zum Video, damit du vor dem Umzug weißt, was mitkommt und was still liegen bleibt.

DIE EINE REGEL, AUS DER ALLES FOLGT

Ob der Weg offen ist, entscheidet nicht die Versionsnummer, sondern was in deinem Store liegt.

SÄULE 1

Tiefes technisches Verständnis

Du weißt, dass Vault nicht der Ort deiner Secrets ist, sondern der Prozess mit dem Schlüssel. Damit erkennst du, was ein Prozesstausch leisten kann und was nicht.

SÄULE 2

Open-Source-Lizenz

OpenBao führt den Stand vor der BUSL unter MPL 2.0 in der Linux Foundation weiter. Ohne diesen Fork gäbe es kein Ziel, zu dem du wechseln könntest.

01 Die Zeitleiste: was wann passiert ist

Video 01:16

WANN	WAS
Aug. 2023	HashiCorp wechselt auf BUSL 1.1, nicht OSI-anerkannt. Letzte Vault unter MPL 2.0: 1.14.8
Dez. 2023	OpenBao startet als Fork in der Linux Foundation, weiter MPL 2.0
Feb. 2025	IBM schließt die Übernahme von HashiCorp ab
Apr. 2026	Vault springt von 1.21 auf 2.0. Zwischen Fork-Punkt und heute liegen acht Releases Storage-Entwicklung

02 Der Prozesstausch, in vier Schritten

Video 05:49

- [ ] Vault stoppen. Das Datenverzeichnis bleibt liegen, es wird nichts exportiert und nichts importiert.
- [ ] In der Konfiguration `disable_mlock` entfernen, die Option gibt es in OpenBao nicht mehr. Das ist die ganze Anpassung.
- [ ] `bao server` auf demselben Verzeichnis starten und mit **denselben Shamir-Anteilen** entsiegeln.
- [ ] Gegenprobe: `bao version-history` führt danach die alte Vault-Version. Das beweist, dass es derselbe Store ist und keine Kopie.

```
bao operator unseal <Anteil 1>
bao operator unseal <Anteil 2>
bao version-history
# 1.14.1 Vault / 2.0.3 Vault / 2.6.1 OpenBao
```

03 Der Inventur-Bogen: vor dem Stoppen

Video 13:31

Diese drei Befehle laufen in **Vault**, solange es noch läuft. Ihr Ergebnis ist die Landkarte deiner Installation, und sie ist reiner Text.

```
vault secrets list
vault auth list
vault policy list
# jede Zeile gegen die Liste in 04 prüfen
```

Jeder Mount, dessen Typ in der Liste unten **nicht** vorkommt, ist ein Kandidat für den stillen Ausfall. Notiere ihn mit Pfad, Typ und Besitzer, bevor du irgendetwas stoppst.

04 Was OpenBao 2.6.1 eingebaut mitbringt

Video 09:22

ART	EINGEBAUT IN OPENBAO
Secret Engines	kubernetes, kv, ldap, openldap, pki, rabbitmq, ssh, totp, transit
Auth-Methoden	approle, cert, jwt, kerberos, kubernetes, ldap, oidc, radius, userpass
Database	cassandra, influxdb, mysql, postgresql, redis, valkey

Vault 2.0.3 bringt 19 Secret Engines und 18 Auth-Methoden mit, OpenBao je 9. Schau nicht auf die Anzahl, schau auf die Namen: Was fehlt, heißt `aws`, `azure`, `gcp`, `alicloud`, `oci`, `okta`. Das Herstellerspezifische ist ausgelagert, es gibt es weiterhin als externes Plugin.

05

Video 09:22

Das stille Fehlerbild erkennen

Ein Mount ohne Plugin bricht den Start **nicht** ab. Er wird übersprungen und steht danach trotzdem in der Liste:

```
# im Startlog von OpenBao:
[ERROR] failed to create mount entry: path=aws/
[WARN]  skipping plugin-based mount entry: path=aws/
# im Betrieb, und das ist die Falle:
bao list aws   ->   No value found at aws
```

`No value found` ist derselbe Satz, den auch ein leerer Pfad liefert. Ehrlich wird die Antwort erst beim Schreiben: `route entry found, but backend is nil`. Wer nur liest, sieht den Unterschied nicht.

06

Video 13:31

Die drei Töpfe: sortier deinen Bestand

TOPF	WAS HINEINGEHÖRT	WAS DU TUST
Landkarte	Engines, Auth-Methoden, Policies, Rollen	in Code neu aufbauen, nicht kopieren
Dynamisch	Datenbank-Zugänge, Zertifikate, Cloud-Credentials	nichts. Sie entstehen im Zielsystem von selbst neu
Statisch	KV-Werte, Root-Credentials, CA-Schlüssel	rotieren, nicht umziehen

Der dritte Topf ist die einzige echte Arbeit. Und er ist zugleich das Maß deines Lock-ins: Was du nicht rotieren kannst, hält dich fest, nicht der Store.

07

Video 13:31

Cutover-Reihenfolge, stückweise statt Urknall

- [ ] **Inventur und Abgleich** (Sektion 03 und 04). Ergebnis ist eine Liste von Mounts, die drüben fehlen.
- [ ] **Tote Mounts abräumen oder ersetzen.** `bao secrets disable` beziehungsweise `bao auth disable`, oder das passende externe Plugin installieren.
- [ ] **Landkarte in Code gießen**, mit OpenTofu gegen die neue Instanz. Dann ist der Wechsel keine Migration, sondern dieselbe Definition auf einem anderen Ziel.
- [ ] **Statischen Bestand rotieren**, Verbraucher umstellen, erst danach die alte Instanz abschalten.

08

Video 15:42

Die Prüffrage, die zählbar ist

Wenn du deinen Secret-Store morgen ersetzen müsstest: wie viele Werte müsstest du von Hand hinüberretten? Jeder davon ist ein Lock-in, den du selbst gebaut hast.

Die naheliegende Regel „jedes Secret braucht ein TTL“ ist falsch. Ein TTL ist eine Eigenschaft des **Zielsystems**, nicht des Stores: Räumt KV v2 per `delete_version_after` die eigene Kopie weg, bleibt das Datenbankpasswort gültig. **Gelöscht ist nicht rotiert.**

Die tragfähige Fassung: **kein Secret ohne Besitzer und ohne erprobtes Rotationsverfahren.** Das TTL ist die automatisierte Form davon. Wo keins geht, tritt ein dokumentiertes, mindestens einmal durchgeführtes Verfahren mit Termin an seine Stelle.

ZWEI FALLEN, DIE DU SONST SUCHST

Erstens: Ab Vault 2.0 legt jeder Store selbst einen Mount `agent-registry/` an. Dieses Plugin gibt es in OpenBao nicht, du hast also mindestens einen toten Eintrag, ohne eine Entscheidung dazu getroffen zu haben. **Zweitens:** `bao plugin reload` meldet nach dem Nachinstallieren zwar Erfolg, hinterlässt den Mount aber halb kaputt (`cannot write to storage during setup`). Es hilft nur ein Neustart.

Die Lock-in-Frage lautet nicht, ob du deine Werte mitnehmen kannst. Sie lautet, ob du sie wegwerfen kannst.

Dieser Spickzettel gehört zum Video über Vault, OpenBao und die Frage, was ein Prozesstausch wirklich mitnimmt. Jede Woche praxisnahe Antworten auf Docker- und Kubernetes-Probleme: der Newsletter.

trutz.io

Newsletter