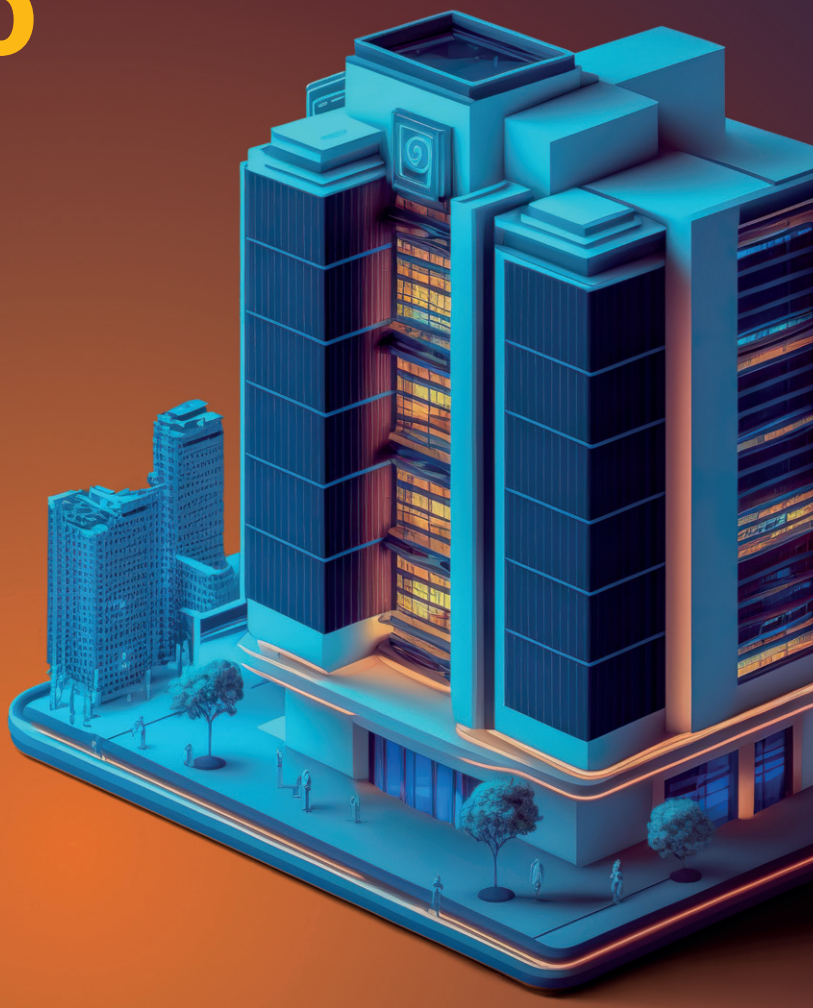
01 | 26 Sonderdruck für Adfinis

IT Administrator
Das Magazin für professionelle System- und Netzwerkadministration

Secrets-Management mit OpenBao



www.it-administrator.de

Secrets-Management mit OpenBao

Nicht weitersagen!

von Markus Feilner


Quelle: Vladimirfloyd – 123RF

Eine Schatzkammer oder den Banksafe schützen massive Wände oder dicke Türen mit hochsicheren Schlössern. Und oft sind mehrere Schlüssel und mehrere Personen erforderlich, um dieses Schloss zu öffnen. Ein vergleichbares Sicherheitsniveau für digitale Geheimnisse wie Credentials, Token oder SSH-Zugänge bringt die Open-Source-Software OpenBao in die Unternehmens-IT. Wie die Software Secrets schützt und verwaltet, zeigt unser Workshop.

OpenBao [1] entstand als Fork, als HashiCorp seinem Secrets Manager "Vault" den Open-Source-Status nahm und die Software unter die Business-Source-Lizenz stellte. Dadurch wurde der freie, kommerzielle Einsatz nur sehr eingeschränkt möglich und Cloudanbieter ohne Lizenz ausgesperrt. OpenBao steht unter den Fittichen ehemaliger HashiCorp-Community-Mitglieder, aber auch IBM und anderen großen Unternehmen. Ursprünglich ein Fork von Vault 1.14, dem letzten Release unter der MPL 2.0, verwaltet heute die OpenSSF OpenBao als Sandbox-Projekt.

Geheimnisse flexibel verwaltet

OpenBao ist ein Tool zum sicheren Verwalten von Geheimnissen (Secrets). Der Funktionsumfang des Originals und seines Forks ist heute sehr ähnlich, beide speichern sensible Daten wie Passwörter, API-Schlüssel, Zertifikate, Tokens oder Datenbank-Zugangsdaten, kontrollieren die Verwendung und zentralisieren, automatisieren und schützen deren Ausgabe.

Ein Secret kann dabei alles sein, was nicht öffentlich sein sollte, also Passwörter, API-Schlüssel, TLS-Zertifikate, SSH-Zugänge, Zugangsdaten zu Datenbanken oder auch Tokens für Cloudanbieter. Seine Dienste bietet OpenBao über ein Webinterface, ein CLI oder ein API an. Für das API kommen Agenten oder eigene Skripte auf Clients zum Einsatz, das Webinterface läuft in allen modernen Browsern. Ein Developer Mode erleichtert den Einstieg und das Lernen.

Alle Geheimnisse lagert das Tool in einem verschlüsselten Speicher (Secret Storage) und über "Dynamic Secrets" erzeugt es Zugangsdaten beispielsweise für Datenbanken oder Clouds – auch on-demand und mit automatischem Ablaufdatum. Richtlinien beziehungsweise Access Control Lists (ACLs) regeln mit feingranularer Kontrolle, wer auf welche Secrets zugreifen darf. Das Audit-Logging zeichnet alle Zugriffe auf, um Compliance- und Sicherheitsanforderungen abzudecken. Darüber hinaus kann OpenBao als API-basierter Verschlüsselungsdienst dienen (Encryption-as-a-Service), dies spart unter Umständen viel Crypto-Code in eigenen Anwendungen.

OpenBao ist sehr flexibel: Die generierten Secrets können nach einer definierten Zeit ablaufen (Time To Live, TTL), oder automatisch widerrufen werden (Secret Leasing & Revocation). Für Anwendungen, die eine Datenbankverbindung benötigen, lassen sich temporäre Zugangsdaten bereitstellen und für den Zugriff auf Cloudressourcen wie AWS generiert OpenBao kurzlebige IAM-Tokens. Für Entwicklerteams, die Zugang zu verschiedenen APIs brauchen, verwaltet OpenBao die Keys zentral, genau wie in der Zusammenarbeit mit einer Public Key Infrastructure (PKI). Auch deren Zertifikate sollen schließlich automatisch erstellt, verteilt und erneuert werden.

Sichere, temporäre Zugangsdaten

Um beispielsweise einer Applikation temporären Zugang zu einer Datenbank zu erlauben, ohne feste Zugangsdaten im Code oder in einer Config-Datei speichern zu müssen, geht OpenBao wie folgt vor: Eine App bittet den Server um Zugang zur Datenbank. Das Tool generiert nun dafür temporäre Zugangsdaten, die beispielsweise für eine Stunde gültig sind und übermittelt diese an den Client. Der hat nun 60 Minuten Zeit, danach ist der Login ungültig.

Weil hierbei keine festen Credentials mehr nötig sind, kein Entwickler die eigentlichen Zugangsdaten sieht und alle kompromittierten Daten automatisch verfallen, erhöht das Vorgehen die Sicherheit im Unternehmen massiv. Dafür verschlüsselt OpenBao alle Daten im Speicher und verwendet SSL/TLS während der Übertragung. Außerdem bietet es einen "Seal/Unseal"-Mechanismus: Beim Start muss jeder Storage zunächst "entsiegelt" werden – analog zu einem Hochsicherheitssafe in einer Bank braucht es zum Entsiegeln (also quasi dem "Öffnen" oder "Aufschließen" des Speichers) einen oder mehrere Schlüssel, deren Verwendung Audit-Logs ebenso protokollieren wie jeden Zugriff auf ein Secret.

OpenBao arbeitet dafür mit Tokens, die mit einer Richtlinie verknüpft sind. Jede davon ist pfadbasiert, Regeln schränken

die Aktionen und den Zugriff auf die Pfade ein. Mit OpenBao lassen sich Tokens manuell erstellen und zuweisen; auch ein Self-Service, an dem sich Clients anmelden und ein Token erhalten, ist möglich. Gleichwohl ist es aber eigentlich vorgesehen, sogenannte "Auth Methods" zu benutzen, womit sich die Clients sauber integrieren, identifizieren und bei Bedarf revozieren lassen.

Der Workflow von OpenBao besteht dabei aus vier Phasen: Sobald der Client anhand einer Authentifizierungsmethode bestätigt wurde, entsteht ein Token mit einer zugeordneten Richtlinie. Dazu validiert OpenBao den Client (Mensch oder Maschine) typischerweise anhand vertrauenswürdiger Quellen wie Identity-Provider (OIDC oder LDAP). Der Client nutzt dann dieses Token, um eine Abfrage zu starten. Ist der Abgleich mit der OpenBao-Sicherheitsrichtlinie erfolgreich, gelingt dem Client der Zugriff auf die Secrets.

Die Richtlinien sind eine Reihe von Regeln, die festlegen, auf welche API-Endpunkte ein Client mit seinem OpenBao-Token Zugriff hat. Sie gewähren oder verbieten den Zugang zu bestimmten Pfaden und Vorgängen in OpenBao, auf Geheimnisse, Schlüssel und Verschlüsselungsfunktionen. Der Secret Manager stellt dafür ein Token aus, das mit der Identität des Clients und den verbundenen Richtlinien in Einklang steht und das dieser in Zukunft verwenden darf.

OpenBao installieren

Für einen Test von OpenBao haben Sie verschiedene Möglichkeiten: Die Software ist in vielen Standard-Repositories enthalten, es gibt Container-Images in den gängigen Registries, vorkompilierte Binaries, aber auch die Installation aus Quelltext oder via Helm Charts ist vorgesehen [2]. Egal wie, meist läuft die Installation nach dem gleichen Muster ab: Binaries (oder Images) holen oder erzeugen, Signaturen prüfen OpenBao installieren und entsiegeln (hier taucht der Unseal Key auf) und mit der Konfiguration loslegen. Danach gilt es noch, für das automatische Starten und Entsiegeln des OpenBao-Servers zu sorgen.

Je nach Einzelfall sind bei der Konfiguration zahlreiche Schritte zu erledigen, die individuell unterschiedich ausfallen: Serveradresse, Port und die UI-Einstellungen, Speicherbackend, API-Einstellungen (API-Adresse und Port), TLS-Zertifikate, aber auch Protokollierungseinstellungen wie die Genauigkeit (Verbosity) der Logs und deren Speicherort sind festzulegen. Dazu kommen Einstellungen, die Authentifizierung und Sicherheit festlegen, beispielsweise für das Ver- und Entsiegeln und die Revozierung des "Root Tokens".

Viele Organisationen betreiben OpenBao als einen Verbund von hochverfügbaren OpenBao-Servern. In den meisten Fällen arbeiten sogar gleich zwei bis drei solcher Cluster als "Spielwiese", Integrations- oder als Produktionsumgebung.

Für den schnellen Start unter Linux bietet es sich an, OpenBao aus dem Standard-Repository zu installieren, beispielsweise auf OpenSUSE mit *zypper in openbao*. Das Kommando *bao -h* zeigt zum Einstieg die Befehlsvielfalt des CLI – der bao-Befehl agiert sowohl zur Steuerung des Servers, übernimmt aber auch viele Funktionen eines Clients. Zu jedem der über den Parameter "-h" gelieferten Einträge zeigt ein erneutes Hilfekommando weitere Details.

Einen Development-/Test-Server (mit temporären Inhalten) starten Sie via *bao server -dev*, wobei der Export der Variable "BAO_ADDR" dem Tester später lästige Tipparbeit spart. Mit:

```
export BAO_ADDR='http://
  localhost:8200'; bao server -dev
```

starten Sie den Server und erhalten sowohl ein Root-Token als auch einen Unseal-Key. Ab sofort liegt das Webinterface auf Port 8200 des lokalen Rechners bereit und Sie starten es im Browser über die Adresse "http://localhost:8200". Aber Achtung: Alle Daten dieser Instanz sind temporär und flüchtig – nach einem Stop und Neustart des bao-Kommandos steht wieder ein leerer Server zur Verfügung. Im Web-GUI füllen Sie das Eingabefeld "Token"mit Ihrem "Root-Token". Nach der Anmeldung finden Sie sich auf der Seite "Secrets Engine" wieder und verfügen über einen vollständigen, aber noch recht leeren OpenBao-Server.

Noch ein Hinweis zum Root-Token: Dieses erlaubt privilegierten und anonymen Zugriff auf OpenBao und ist nur für die initiale Konfiguration sowie für Notfälle ("Break Glass") vorgesehen. Administratoren sollten immer mit persönlichen und eingeschränkten Benutzeraccounts arbeiten. Dabei bedeutet "Least Priviledge Access" hier, dass auch Admins zwar die Plattform betreiben können, aber keinen Zugriff auf die Secrets der Teams haben.

Versiegeln und Entsiegeln

Den Versiegelungsmechanismus ("Seal") erreichen Sie über die Schaltfläche "Seal OpenBao" – den Unseal-Key haben Sie ja beim Setup bereits erhalten. Das Versiegeln einer Instanz weist den OpenBao-

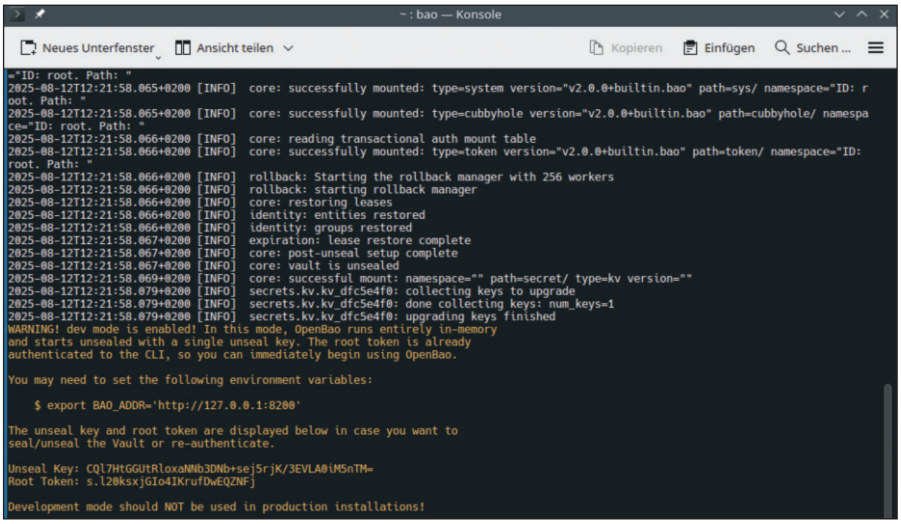


Bild 1: Der erste OpenBao-Server ist gestartet. Dieser ist noch temporär und somit löscht ein Neustart alle Daten.

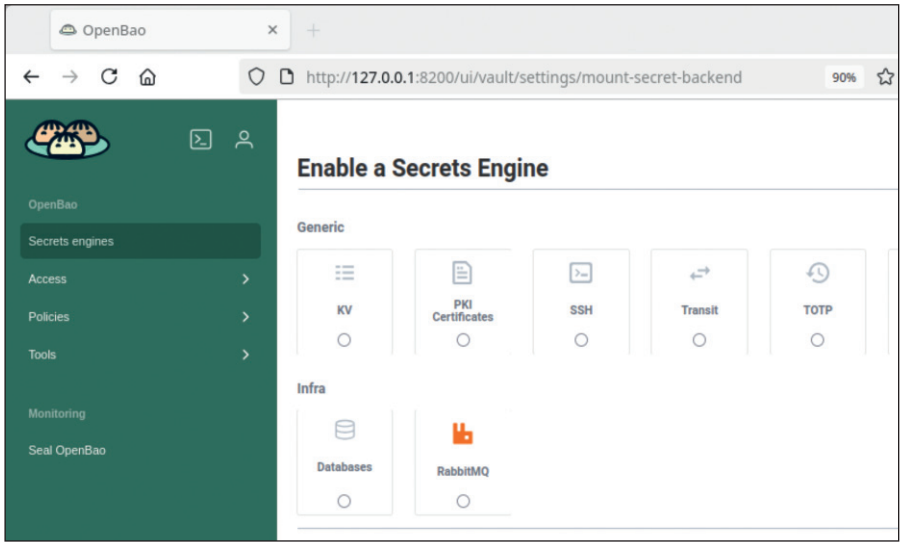


Bild 2: OpenBao bietet eine Auswahl nutzbarer Secrets Engines im Webinterface.

Server an, nicht mehr auf Zugriffsvorgänge zu reagieren, bis die Versiegelung wieder aufgehoben wird. Erst wenn Sie den Tresor mit dem Unseal-Befehl (CLI, Webinterface oder über das API) wieder öffnen, ist der Zugriff erneut möglich. Normalerweise starten OpenBao-Server in diesem versiegelten Zustand. Die Software weiß dann zwar, wo ihre Daten liegen, welches Storage-Backend sie verwenden soll und so weiter, kann die Secrets aber nicht entschlüsseln. Beim Dev-Server in unserem Beispiel passiert das Versiegeln und Entsiegeln für Lern- und Testzwecke automatisch.

Die Standardkonfiguration von OpenBao verwendet ein Shamir-Siegel: Der Entsiegelungsschlüssel wird nicht direkt an einen oder mehrere Anwender verteilt, sondern nach dem "Shamir's Secret Sharing"-Algorithmus [3] in Teile gestückelt. Eine bestimmte Anzahl von Teilen ("Threshold") ist erforderlich, um den Entsiegelungsschlüssel zu rekonstruieren, der dann zur Entschlüsselung des Haupt-Keys dient. Die Anteile des Schlüssels werden dafür nacheinander (in beliebiger Reihenfolge) hinzugefügt, bis genügend von ihnen vorhanden sind, um den Key zu rekonstruieren. Der Shamir-Prozess ("Key Ceremony") kommt im Übrigen auch zum Einsatz, um ein Root-Token für den Notfallzugriff zu erzeugen. Für produktive Umgebungen kommt aber normalerweise "Auto-Unseal" ins Spiel, damit Server nach einem Reboot funktionsfähig sind.

Eine große Hilfe ist dabei das bao-operator-Kommando, das eine Reihe von Operationen für den Betrieb und die Verwaltung ermöglicht:

- *bao operator init*: Initialisiert einen neuen OpenBao-Cluster, indem ein Root-Schlüssel generiert und das Storage-Backend vorbereitet wird.
- *bao operator unseal*: Entsiegelt einen OpenBao-Server.
- *bao operator seal*: Versiegelt einen OpenBao-Server, wodurch dieser keine Operationen mehr ausführen kann.
- *bao operator rekey*: Erzeugt einen neuen Satz von Unseal-Schlüsseln.
- *bao operator validate-config*: Validiert die OpenBao-Konfiguration.
- *bao operator diagnose*: Ermöglicht die Diagnose von Problemen mit OpenBao.

Und haben Sie ihr Root-Token verloren und sich so unvermittelt ausgesperrt, erzeugen Sie via *bao operator generate-root* ein neues. Den Erfolg oder Misserfolg Ihrer Eingaben prüfen Sie mit *bao status*. Alle Kommandos erklärt die ausfürliche Onlinedokumentation [4].

Secrets Engine einrichten

Damit OpenBao seine Dienste verrichten kann, benötigt es eine Secrets Engine, also ein Verfahren zum Speichern von Geheimnissen. Im Developer-Mode-Server sind zwei aktiv: Cubbyhole, ein "Per-token private Secret Storage" und "secret", ein Key/Value-Secret-Storage" (im OpenBao-Kontext meist abgekürzt mit "KV"). Klicken Sie im Hauptmenü des Browser-Interface auf "Secrets Engines", gelangen Sie zur Auswahlmaske "Enable a Secrets Engine". Dort sind Sie in der Lage, unter anderem KV-Storage, PKI-Zertifikate, SSH, Datenbanken, RabbitMQ, Kubernetes oder auch zeitlich limitierte One-Time-Passworte (TOTP) zu aktivieren. Das Gleiche gelingt über das CLI mit *bao secrets enable ssh*. Der Server meldet dann "Success! Enabled the ssh secrets engine at: ssh/" und zeigt im Webinterface einen dritten Eintrag für SSH bei den Secrets Engines. Jetzt gilt es, das Backend zu konfigurieren und eine Rolle für SSH zu erzeugen, wobei auch OTP möglich ist.

Beachten Sie jedoch, dass Sie in jedem Fall auch die Clientsysteme konfigurieren müssen. Mit SSH kann der OpenBao-

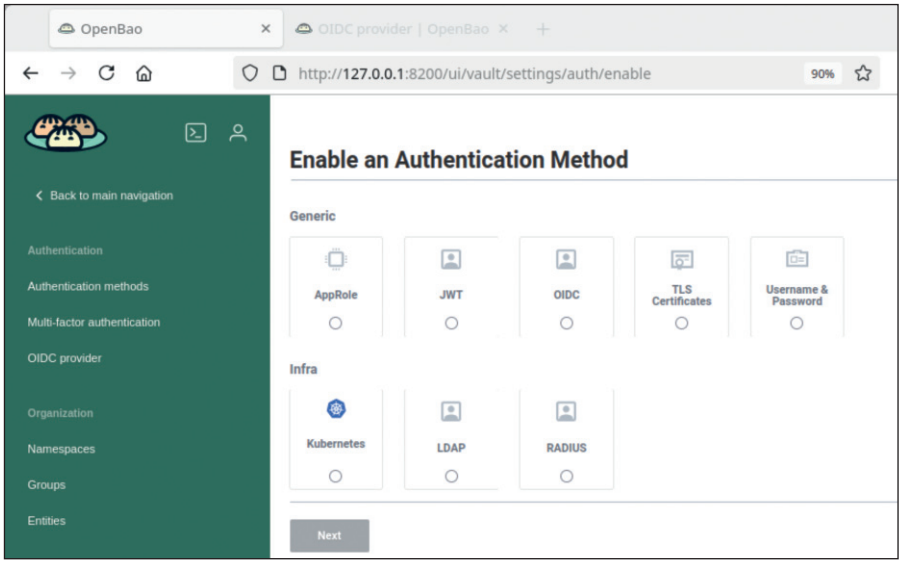


Bild 3: Und auch bei den verfügbaren Authentifizierungsmethoden öffnet OpenBao seinen umfangreichen Bauchladen.

Server sowohl One-Time-Passwörter als auch PKI-Funktionen nutzen, etwa wenn der SSH-Key durch eine CA signiert sein soll und nur dann Zugang erhalten darf. Das ist mit OpenBao möglich, erfordert jedoch entsprechende Einstellungen auf den Clients, beispielsweise in der Konfiguration von SSH und PAM. Die SSH-CA-Methode mit OpenBao ist hier zu bevorzugen, weil sie gleichermaßen elegant und sicher temporäre SSH-Zugänge gewährt, ohne dauerhaft Public-Keys oder Passwörter auf Servern hinterlegen zu müssen.

OpenBao signiert dabei (temporär) die öffentlichen Schlüssel, die die Clients für die Authentisierung bei SSH-Servern verwenden. Der Server akzeptiert nur derart signierte Keys, wenn er der CA vertraut [5]. Der Admin muss dazu auf dem Server den CA-Public-Key als vertrauenswürdig hinterlegen und ihn auf dem Zielserver in die Datei "/etc/ssh/trusted-user-ca-keys.pem" einfügen. Den CA-Public-Key holt er dafür (mit Root-Rechten) aus OpenBao:

```
bao read -field=public_key ssh/con-
  fig/ca > trusted-user-ca-keys.pem
```

Dann setzen Sie in der SSHD-Konfiguration im File "/etc/ssh/sshd_config" die folgende Werte:

```
TrustedUserCAKeys /etc/ssh/
  trusted-user-ca-keys.pem
PubkeyAuthentication yes
```

Anschließend erfolgt ein Neustart mit *sudo systemctl restart sshd*. Jetzt können Sie die Verbindung mit dem von OpenBao bereitgestellten Key testen – wenn alles klappt, kann er sich auf dem Zielserver anmelden, ohne dass dort der Client-Key jemals gespeichert war. Eine Widerrufsliste (Revocation List) müssen Sie an dieser Stelle allerdings manuell pflegen.

OpenBao mit Gitlab-CI-Pipelines

OpenBao bringt einen eigenen OpenID-Connect-(OIDC)-Provider [6] mit, bietet Rollen sowie Policies und diverse Authentifizierungsmethoden [7] inklusive Multifaktor, OTP, Namespaces, Access Control Lists und gruppenbasierten Regeln. Unsere

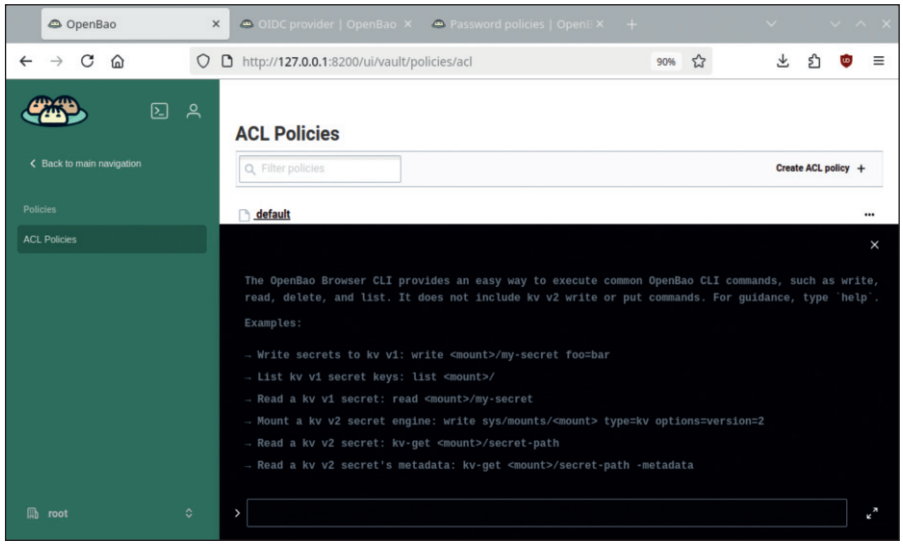


Bild 4: Innerhalb des Webinterface lässt sich parallel eine CLI-Konsole aufrufen.

beiden SSH-Beispiele sollten nur als Startpunkte für das Verständnis von OpenBao dienen, denn die Software erlaubt auch die Integration mit komplexen Mechanismen. Bei vielen entsprechenden Anleitungen aus dem Internet ist allerdings Vorsicht geboten. Denn diese "hacken" per Skript den Seal/Unseal-Mechanismus oder verwenden sogar Root-Tokens. Solche Vorgehensweisen sollten Sie maximal zum Testen, keinesfalls aber in Produktivumgebungen einsetzen.

Um Entwicklungs- und Deployment-Schritte zu automatisieren, benutzen viele Organisationen heutzutage so genannte Pipelines. Und weil die Integration von GitLab-CI-Pipelines nahtlos und sicher mit OpenBao funktioniert, hat sie sich unter IT-Verantwortlichen etabliert. Bei jedem Pipeline-Durchlauf erzeugt dabei die GitLab-Platform ein JSON Web Token (JWT), das wiederum die starke Authentifizierung durch OpenBao erlaubt. Um eine solche Pipeline zu integrieren, müssen Sie zunächst in OpenBao "OIDC" als Authentifizierungsmethode aktivieren und konfigurieren. Im Zusammenspiel von Rollen und Policies lässt sich der Zugriff für GitLab-Gruppen, -Projekte und -Pipelines feingranular gestalten. Als letzten Schritt müssen Sie die Pipeline um Instruktionen ergänzen, um sie mit dem JWT bei OpenBao anzumelden und die gewünschten Secrets zur Laufzeit auslesen zu dürfen [8]. OpenBao spricht fortan transparent im Hintergrund mit der GitLab-Plattform um JWTs zu verifizieren und zu prüfen, ob der angeforderte Zugriff erlaubt ist.

Fazit

OpenBao ist ein mächtiges und überaus komplexes Werkzeug. Trotz umfangreicher Dokumentation und agiler Community bleibt die Lernkurve jedoch steil, und verirrt anfangs mit einer Vielzahl an Konfigurationsoptionen, Befehlszeilen und Features. Da helfen auch das aufgeräumte Webinterface und das übersichtliche CLI nicht – zu groß ist der Funktionsumfang. Das aber zeugt gleichzeitig von der Flexibilität der Software, die dieser Workshop vorgestellt hat. Dergestalt lassen sich moderne Anforderungen an das Secrets-Management mit einer freien Software umsetzen. *(jp)* IT

Links

[1] **OpenBao**
it-a.eu/pap21
[2] **OpenBao installieren**
it-a.eu/pap22
[3] **Shamir's secret sharing**
it-a.eu/pap23
[4] **OpenBao CLI-Referenz**
it-a.eu/pap24
[5] **Signierte SSH-Zertifikate**
it-a.eu/pap25
[6] **OIDC-Provider**
it-a.eu/pap26
[7] **Auth-Methoden**
it-a.eu/pap27
[8] **JWT/OIDC**
it-a.eu/pap28