Dynamic Credentials

Nils Bokermann, München 09.11.2023

Über mich

Nils Bokermann

- freiberuflicher Softwareentwickler
- Chemiker ;-)

Schwerpunkte

- Entwicklung von Java Enterprise Anwendungen
- Ende zu Ende Verantwortlichkeit


info@bermuda.de


@sanddorn


xing.to/sanddorn

Agenda

- Begriffsklärungen
- Risiken und Mitigation
- 3 exemplarische Szenarien
- Gelöste/Ungelöste Probleme

Begriffe

- Authentisierung (neben Authentifizierung) (engl. authentication):
  Bestätigung der Identität einer Person oder eines Systems

- Autorisierung (engl. authorisation):
  Erteilung von Rechten/Rollen an eine Person oder ein System

Was sind Credentials

- Credentials sind zunächst Identitätsnachweise, in zweiter Linie können es auch Berechtigungsnachweise sein.Wikipedia:

> Ein **Berechtigungsnachweis** (auch Login-Daten, Anmeldedaten, Anmeldeinformationen oder englisch *credential*) ist ein Instrumentarium, das einem System die Identität eines anderen Systems oder eines Benutzers bestätigen soll.
>
> **Wikipedia**

Welche Credentials?

Natürliche Personen

2FA

Systeme

Username
Password

API-Token

Zertifikat
Privater Schlüssel

Risiken statischer Credentials

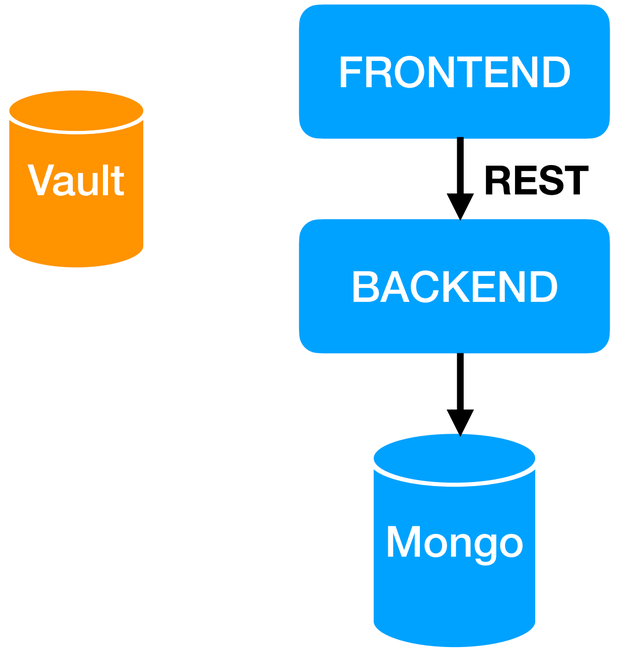
- Diebstahl/Verlust
- Ausspähen
- Angriffe auf die User-Datenbank (z.B. Rainbow-Attack)
- Credential-Sharing

Risk Mitigation

- Need-To-Know-Prinzip
- Credential-Rotation
- One-Time-Password

Systemarchitektur

- Zusätzliches System: Hashicorp Vault

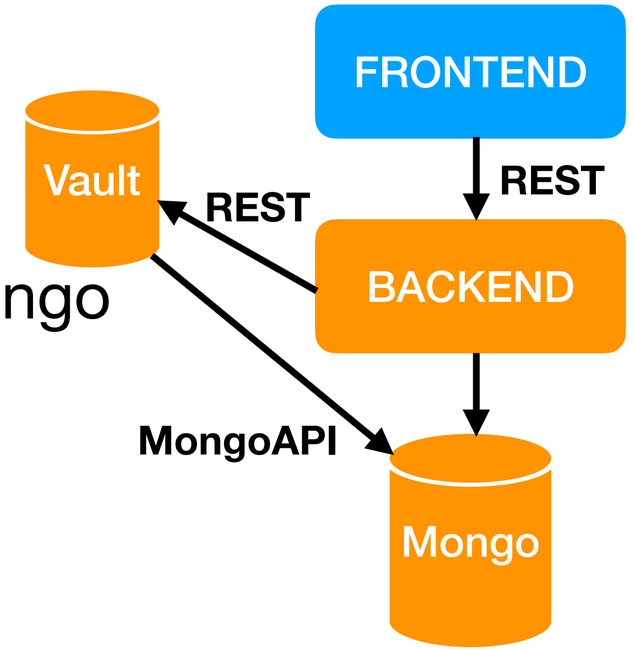


Datenbank-Credentials

- DBA nicht mehr involviert
- Übergabe eines Teils der DBA-Verantwortlichkeit an ein System!
- Credentials brauchen nicht mehr im File-System gespeichert sein

Datenbank-Credentials

- Vault-Konfiguration:
  - MongoDB-Connection mit DBA-Rechten
  - „Role“ mit dem die User-Rechte auf der Mongo gesteuert werden
- Spring-Applikation:
  - `application.properties`
  - Lease-Renewal über einen `LeaseListener`



application.properties

```
spring.cloud.vault.database.backend=database
spring.cloud.vault.database.enabled=true
spring.cloud.vault.database.role=${BACKEND_DB_ROLE:backend}
spring.cloud.vault.database.username-property=spring.data.mongodb.username
spring.cloud.vault.database.password-property=spring.data.mongodb.password
```

LeaseListener

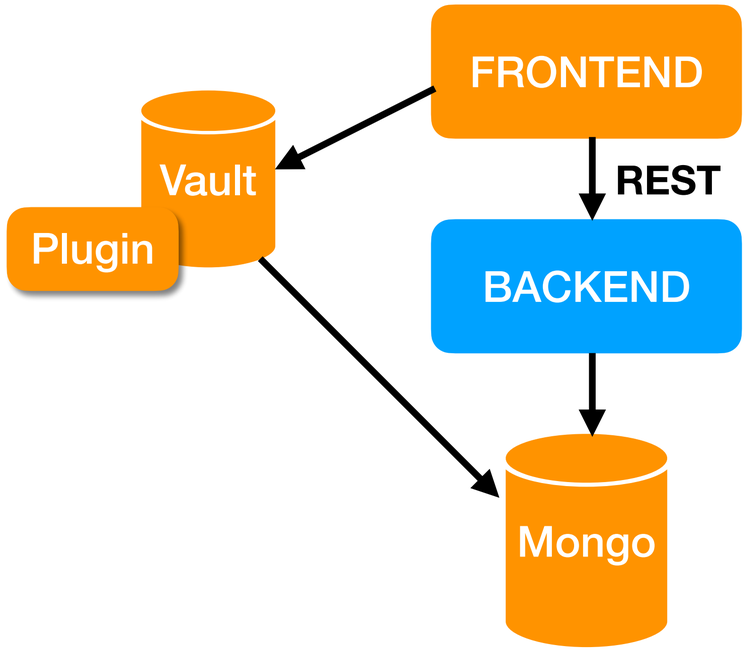
```
@PostConstruct
private void postConstruct() {
    leaseContainer.addLeaseListener(new RotatingLeaseListener());
}

class RotatingLeaseListener implements LeaseListener {
    @Override
    public void onLeaseEvent(SecretLeaseEvent secretLeaseEvent) {
        try {
            if (secretLeaseEvent.getSource().getPath().equals(vaultCredsPath)) {
                LOGGER.info("Lease Change Event for DB");
                if (secretLeaseEvent instanceof SecretLeaseExpiredEvent
                        && secretLeaseEvent.getSource().getMode() == RequestedSecret.Mode.RENEW) {
                    LOGGER.info("Replace RENEW for expired credential with ROTATE");
                    leaseContainer.requestRotatingSecret(vaultCredsPath);
                } else if (secretLeaseEvent instanceof SecretLeaseCreatedEvent
                            secretLeaseCreatedEvent
                        && secretLeaseEvent.getSource().getMode() == RequestedSecret.Mode.ROTATE) {
                    var credentials = retrieveCredentials(secretLeaseCreatedEvent.getSecrets());
                    if (credentials == null) {
                        LOGGER.error("Cannot get updated DB credentials. Shutting down.");
                        applicationContext.close();
                        return;
                    }
                    refreshDatabaseConnection(credentials);
                }
            }
        } catch (VaultException ve) {
            LOGGER.warn("Vault Exception", ve);
        }
    }
}
```

Basic Authentication

- „Rudel Accounts“ werden vermieden
- Klartext-Passworte stehen nicht in einer Config-Datei — nur im Speicher!
- Rotation der Credentials wird erleichtert, z.T. erst so möglich

Basic-Authentication

- Vault:
  - Plugin um Daten in der Mongo zu verwalten
- Frontent-Applikation:
  - Weiterer Lease-Pfad
  - Lease-Rotation wie für die DB



Basic Authentication

```
public static void main(String[] args) {
 var application = new SpringApplication(HeroApplication.class);
 application.addBootstrapRegistryInitializer(
   VaultBootstrapper.fromConfigurer(new FrontendVaultConfigurer()));
 application.run(args);
}
```

Basic Authentication

```
public class FrontendVaultConfigurer implements VaultConfigurer {
    @Override
    public void addSecretBackends(SecretBackendConfigurer configurer) {
        System.out.println("FrontendVaultConfigurer");
        configurer.add(new AuthorizationBackendMetadata());
        configurer.registerDefaultKeyValueSecretBackends(true);
        configurer.registerDefaultDiscoveredSecretBackends(true);
    }

    public class AuthorizationBackendMetadata implements LeasingSecretBackendMetadata {

        private final PropertyTransformer propertyTransformer;

        public AuthorizationBackendMetadata() {
            var propertyNameTransformer = new PropertyNameTransformer();
            propertyNameTransformer.addKeyTransformation("username", "rest.username");
            propertyNameTransformer.addKeyTransformation("password", "rest.password");
            propertyTransformer = propertyNameTransformer;
        }
```

Basic Authentication

```
public class AuthorizationBackendMetadata implements LeasingSecretBackendMetadata {

    private final PropertyTransformer propertyTransformer;

    public AuthorizationBackendMetadata() {
        var propertyNameTransformer = new PropertyNameTransformer();
        propertyNameTransformer.addKeyTransformation("username", "rest.username");
        propertyNameTransformer.addKeyTransformation("password", "rest.password");
        propertyTransformer = propertyNameTransformer;
    }
```

Plugin: Credential Config

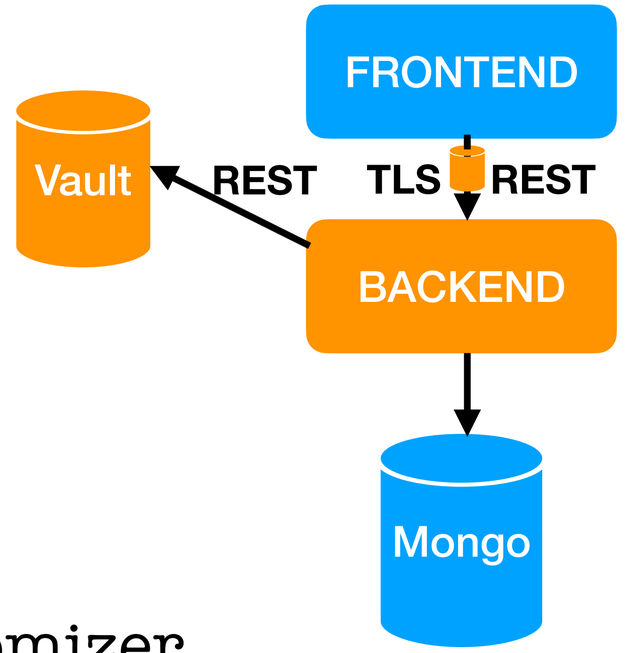
```
func pathCredentials(b *springBootUserBackend) *framework.Path {
    return &framework.Path{
        Pattern: "creds/" + framework.GenericNameRegex("name"),
        Fields: map[string]*framework.FieldSchema{
            "name": {
                Type:        framework.TypeLowerCaseString,
                Description: "Name of the role",
                Required:    true,
            },
        },
        Callbacks: map[logical.Operation]framework.OperationFunc{
            logical.ReadOperation:   b.pathCredentialsRead,
            logical.UpdateOperation: b.pathCredentialsRead,
        },
        HelpSynopsis:    pathCredentialsHelpSyn,
        HelpDescription: pathCredentialsHelpDesc,
    }
}
```

PKI

- Key-Pair automatisiert erstellt
- Private-Key nicht mehr extern auf Platte gespeichert werden.
- Automatisches Re**keying** (nicht nur Re**certification**)

PKI

- Vault-Konfiguration:
  - PKI-Root (CA)
  - „Role“ für die Zertifikate
- Spring-Applikation:
  - Server-Config via `WebServerFactoryCustomizer`
  - Eigener Listener für den Ablauf



TLS Rotation

```
private CertificateBundle issueCertificate() {
    final VaultPkiOperations vaultPkiOperations = vaultTemplate.opsForPki();

    VaultCertificateRequest request = VaultCertificateRequest.builder()
                                                              .ttl(Duration.ofMinutes(15))
                                                              .commonName(commonName)
                                                              .build();

    CertificateBundle certificateBundle;
    try {
        VaultCertificateResponse response = vaultPkiOperations.issueCertificate(pkiRoleName, request);
        certificateBundle = response.getRequiredData();
    } catch (VaultException e) {
        LOGGER.error("VaultException ", e);
        throw e;
    }

    LOGGER.info("Cert-SerialNumber: {}", certificateBundle.getSerialNumber());
    return certificateBundle;
}
```

TLS-Rotation

```
public void customize(TomcatServletWebServerFactory factory) {
        CertificateBundle bundle = issueCertificate();
        KeyStore keyStore = bundle.createKeyStore(ROTATING_KEY);
        String keyStorePath = saveKeyStoreToFile(keyStore);

        ssl.setEnabled(true);
        ssl.setClientAuth(Ssl.ClientAuth.NONE);

        ssl.setKeyStore(keyStorePath);
        ssl.setKeyAlias(ROTATING_KEY);
        ssl.setKeyStoreType(keyStore.getType());
        ssl.setKeyPassword("");
        ssl.setKeyStorePassword(KEYSTORE_PASSPHRASE);
        certificateSerial = bundle.getX509Certificate().getSerialNumber();
        certificateNotAfter = bundle.getX509Certificate().getNotAfter();

        factory.setSsl(ssl);
        factory.setPort(port);
}
```

TLS-Rotation

```java
private void rotateServerKeyInAllConnectors() {
    if (applicationContext instanceof AnnotationConfigServletWebServerApplicationContext) {
        final AnnotationConfigServletWebServerApplicationContext context =
                (AnnotationConfigServletWebServerApplicationContext)
                        VaultCertificateHandler.this.applicationContext;
        final TomcatWebServer webServer = (TomcatWebServer) context.getWebServer();
        Connector[] connectors = webServer.getTomcat().getService().findConnectors();
        Arrays.stream(connectors)
              .filter(connector -> connector.getScheme().equals("https"))
              .forEach(this::rotateConnectorWithRotatingKey);
    }
}
```

Alternativen?

- Gibt es
- **Aber:**
- Framework-Unterstützung
- Dokumentation und Fehlersuche

Ist das alles nötig?

- Short answer: **Nein.**
- **Aber:** Nicht schneller als der Bär, nur schneller als der langsamste.

Tut nichts

Dependency Management

Operative Prozesse

Fragen?

- https://github.com/sanddorn/dynamic-credentials


nils.bokermann@bermuda.de


@sanddorn


xing.to/sanddorn


www.linkedin.com/in/nils-bokermann-3267826

NILS BOKERMANN
IT CONSULTING

- https://www.vaultproject.io/docs
- https://learn.hashicorp.com/collections/vault/custom-secrets-engine
- https://spring.io/guides/gs/vault-config/
- https://geekflare.com/secret-management-software/