Projekt: Vertrauenswürdige Verbindung

Angriffsvektor Connection-Pool

Norman Sibbing (Norman.Sibbing@oracle.com)

BU Core & Cloud Technologies

Oracle Deutschland B.V. & Co KG

Projekt: Vertrauenswürdige Verbindung

Ausgangssituation

- Verarbeitung hochsensibler Daten
- Datenzugriff ausschließlich nur berechtigtes Fachpersonal
- Moderne 3-Schichten-Architektur
- Benutzer Authentifizierung durch Identity Provider (Token)
- Datenbank durch Database Vault und Transparent Data Encryption gesichert
- Risiko: Applikationsserver / Connection-Pool

Projekt: Vertrauenswürdige Verbindung

Risiko: Applikationsserver

- Kommunikation des JDBC Connection-Pools oft unverschlüsselt
- Passwort in Data-Source Konfigurationsdatei

  `<password>geheim</password>` - manchmal sogar verschlüsselt ☺
- Einfache Möglichkeit zum Kapern des Connection-Pools

```
DataSource ds = (DataSource) ctx.lookup("jdbc/HRSESSION");
Connection connection = (Connection)ds.getConnection();
```

- Nutzung eines privilegierten technischen Benutzers

  Reale Benutzer oft nicht in der Datenbank sichtbar

Projekt: Vertrauenswürdige Verbindung

Risikominderung: Kommunikation des JDBC Connection-Pools verschlüsseln

- Verschlüsselung der Kommunikation
  - SQL*Net Nativ
    - Zentral über den Datenbankserver (sqlnet.ora -> SQLNET.ENCRYPTION_SERVER = requested)
    - Die Standardeinstellung Client/Server ist „accepted“ (Gilt auch für Oracle JDBC-Thin)
    - Individuelle Einstellung für JDBC-Thin mittels Connnection-Properties (oracle.net.encryption_client=rejected)
  - SSL/TLS
    - Zertifikate notwendig
    - Datenbank-Listener mit TCPS Endpunkt
    - Individuelle Einstellung für JDBC-Thin mittels Connnection-Properties (javax.net.ssl.keyStore=/data/oracle/wallet/hrapp/cwallet.sso)

Projekt: Vertrauenswürdige Verbindung

Risikominderung: Kein Passwort in der Data-Source Konfigurationsdatei

- Connection-Pool Authentifizierungsmethode
  - Passwort mittels Oracle Secure Enterprise Password Store (SEPS)
    - Passwort im Oracle Wallet gespeichert
    - Passwortänderung komplex
  - Kerberos
    - Kein Passwort vorhanden
    - Authentifizierung über Kerberos-Ticket
  - PKI (TLS / SSO)
    - Kein Passwort vorhanden
    - Authentifizierung über Zertifikat

Projekt: Vertrauenswürdige Verbindung

Risikominderung: Kein Passwort in der Data-Source Konfigurationsdatei

- Connection-Pool Properties für
  - Kerberos

```
oracle.net.encryption_client=requested
java.security.krb5.conf=/etc/krb5.conf
oracle.net.authentication_services=KERBEROS5
oracle.net.kerberos5_cc_name=/data/oracle/kerberos_ticket/krbticket_1000.cc
```

  - PKI SSO (implizite SSL/TLS Kommunikationsverschlüsselung)

```
oracle.net.encryption_client=rejected
oracle.net.authentication_services=TCPS
javax.net.ssl.keyStoreType=SSO
javax.net.ssl.keyStore=/data/oracle/wallet/hrapp/cwallet.sso
javax.net.ssl.trustStoreType=SSO
javax.net.ssl.trustStore=/data/oracle/wallet/hrapp/cwallet.sso
```

Projekt: Vertrauenswürdige Verbindung

Risikominderung: Nutzung eines unprivilegierten technischen Benutzers

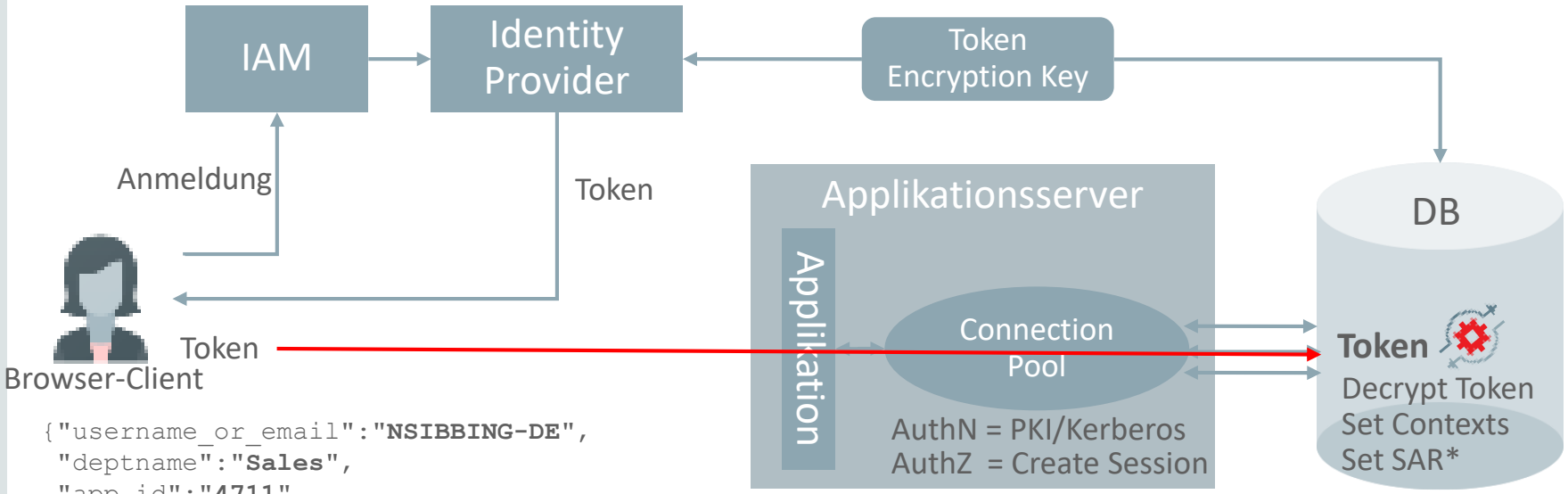
- Connection-Pool verwendet unprivilegierten technischen DB-Benutzer
  - Nur das Create Session Privileg notwendig
- Weitere Vergabe von Privilegien durch Secure Application Roles
- Steuerung der Privilegienvergabe durch entsprechendes Benutzer-Authentifizierungs Token mit kurzer Gültigkeit

```
{"username_or_email":"NSIBBING-DE","deptname":"Sales","app_id":"4711",
"expire_date":"1542015161112","ip_address":"10.200.110.200"}
```

- Missbrauch des Connection-Pools nicht ganz vermeidbar (Restrisiko)
  - Abfangen des Tokens auf dem Applikationsserver

Token AuthZ und AuthN: Schematischer Ablauf

Vertrauenswürdige Verbindung

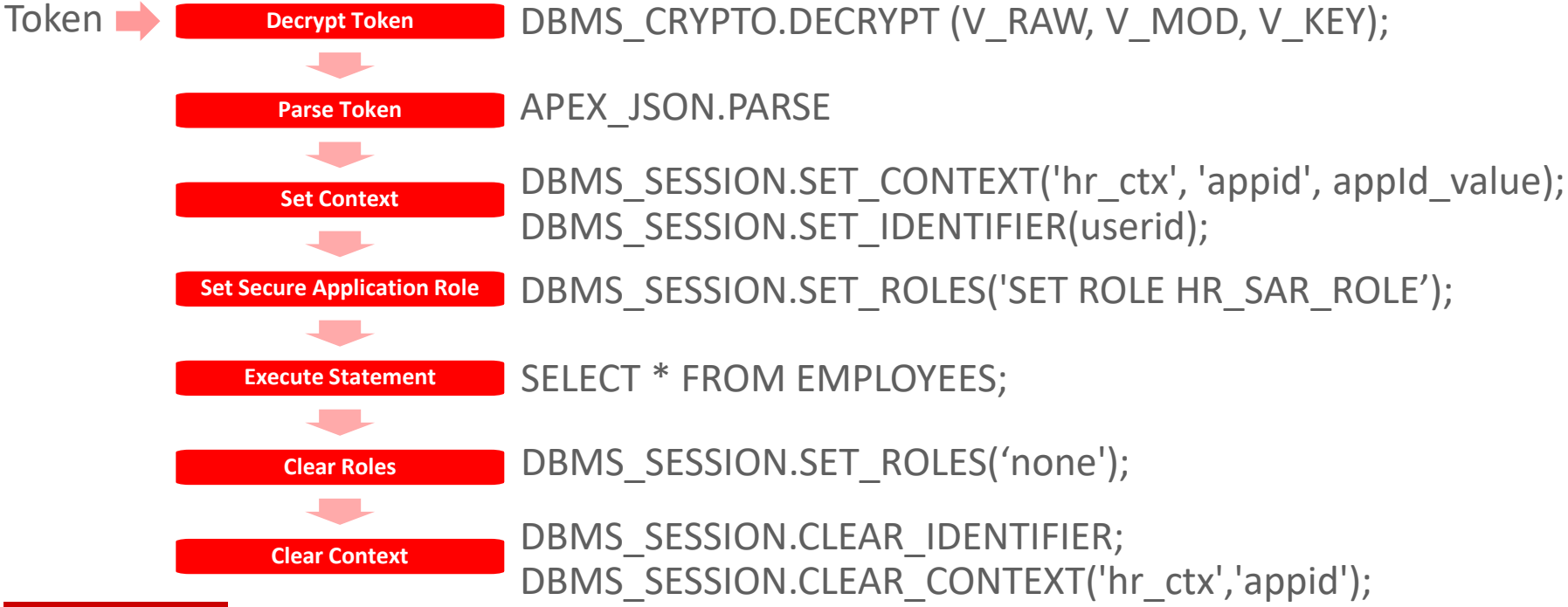



```
{"username_or_email":"NSIBBING-DE",
 "deptname":"Sales",
 "app_id":"4711",
 "expire_date":"1542015161112",
 "ip_address":"10.200.110.200"}
```

* Secure Application Roles

Token AuthZ und AuthN Datenbank-Logik

Vertrauenswürdige Verbindung



Projekt: Vertrauenswürdige Verbindung

Risikominimierung

- Verschlüsselte Kommunikation und Daten
- Connection-Pool ohne Passwort-Authentifizierung
- Datenzugriff nur mit gültigem Token möglich
- Token wird nur im Datenbank-Server entschlüsselt
- Token beinhaltet wichtige Kontext-Informationen z.B. realen Benutzer
- Session-basierte Berechtigungsvergabe gemäß des Tokens
- AuthZ und AuthN-Logik durch Database Vault vor Manipulation geschützt
- Viele weitere AuthZ/N Varianten denkbar (TOTP über Google Authenticator)

Q & A

Connect With Us

/OracleDatabase

/OracleSecurity

blogs.oracle.com/
SecurityInsideOut

Oracle Database Insider

/Oracle/database

/OracleLearning

oracle.com/database/security

oracle.com/technetwork/database/security

Hardware and Software
Engineered to Work Together


ORACLE®