REST, PKI & Oracle DB – klappt das?

Nico Henglmüller
Sphinx IT Consulting
Aspernbrückengasse 2, Wien

Schlüsselworte
PKI, REST, utl_http, JSON

Einleitung

Der sichere Austausch von sensiblen und geschäftskritischen Daten über das Internet ist ein elementarer Bestandteil der Industrie 4.0. Die Anbindung von standardisierten Schnittstellen wie beispielsweise REST Webservices über HTTP ist seit Jahrzehnten fest in die Oracle Datenbank integriert. Als Basis für den sicheren Abruf von Daten ist eine vertrauenswürdige, verschlüsselte Verbindung zum Quellsystem erforderlich. Die notwendigen Werkzeuge bringt Oracle Database in Form der Oracle Wallet als Zertifikatsspeicher und dem Paket utl_http für den Konsum der RESTful Webservices mit.
Einleitend geht dieser Vortrag auf die Motivation und den Nutzen der gewählten Lösung ein. Anschließend werden die Grundzüge einer Public Key Infrastructure und der Ablauf einer TLS gesicherten, durch Clientzertifikate authentifizierten Anfrage erläutert. Abgerundet wird die Präsentation durch ein praktisches Beispiel, um schließlich mit einer Diskussion der Ergebnisse abzuschließen.

Motivation

Oracle stellte mit dem ersten Release der 12c Datenbank umfassende und mächtige Werkzeuge zur Verwaltung von JSON (javascript object notation) Dokumenten in der Datenbank bereit. Dies ermöglicht ein schnelles und komfortables abfragen von nicht relational organisierten Daten durch bewährte SQL Queries. In Kombination mit dem PL/SQL Package utl_http lassen sich Webservices auf JSON Basis kinderleicht abfragen und in die Datenbank einpflegen. In der Realität benötigen Projekte bei dem Austausch von hochsensiblen personenbezogenen Daten einen verschlüsselten Transport und die Authentifizierung des Zugriffs. Durch den Einsatz offener Standards wie HTTP, TLS und die Authentifizierung durch Clientzertifikate ist der Austausch der Daten, direkt aus der Datenbank, gesichert.

Public Key Infrastructure

Die Public Key Infrastructure besitzt eine Hierarchie von Zertifizierungsstellen (engl. Certificate Authority, kurz CA), beginnend bei der Wurzel über intermediate CAs bis zur Verantwortlichen CA der Domäne oder issuing CA. So entsteht eine Kette des Vertrauens von der Wurzel bis zum Client- oder Serverzertifikat. Befindet sich die ausstellende CA innerhalb einer solchen Kette, kann jeder das Zertifikat auf seine Vertrauenswürdigkeit, Echtheit und Gültigkeit überprüfen. Ein Beispiel hierfür ist das grüne Schloss innerhalb eines Browsers in der Adressleiste, wenn Sie die Webseite Ihrer Hausbank öffnen. Dies ist ein Zeichen für den Endbenutzer, dass es sich bei der geöffneten Webseite um eine verschlüsselte und vertrauenswürdige Verbindung handelt.

Um ein Zertifikat zu erzeugen erstellt der Client in einem ersten Schritt ein Schlüsselpaar bestehend aus privatem und öffentlichen Schlüssel und legt die Schlüssel in einen Key Store. Diese Aufgabe übernimmt auf der Datenbank der Oracle Wallet Manager (kurz OWM), welcher die Schlüssel für den Benutzer transparent erzeugt. Anschließend stellt der Benutzer innerhalb des Oracle Wallet Managers einen Certificate Signing Request, in welchem sich der öffentlichen Schlüssel des OWM befindet. Dieser wird an die Zertifizierungsstelle (siehe Abb. 1 Schritt 1) übermittelt, um ein signiertes Clientzertifikat zu erhalten. Die Zertifizierungsstelle erstellt daraufhin ein Clientzertifikat und legt dieses in ein Register.

Das durch die issuing CA erstellte Clientzertifikat wird im Wallet Manager importiert (Schritt 2 in Abb. 1). Im nächsten Schritt prüft dieser den öffentlichen Schlüssel und ordnet das Zertifikat dem zuvor erzeugten Schlüsselpaar zu. Falls sich die Oracle Datenbank, wie in Abbildung 1 dargestellt, in einer Kette aus Zertifizierungsstellen befindet, ist die Hierarchie der Zertifikate ebenfalls innerhalb des Wallet Managers als „trusted certificates“ zu importieren. Im Gegensatz zur Oracle Datenbank handelt es sich bei dem Webserver um einen Service, welcher ein Server Zertifikat mit einem ähnlichen Ablauf bei der issuing CA erfragt und das Zertifikat für den Webservice verwendet.

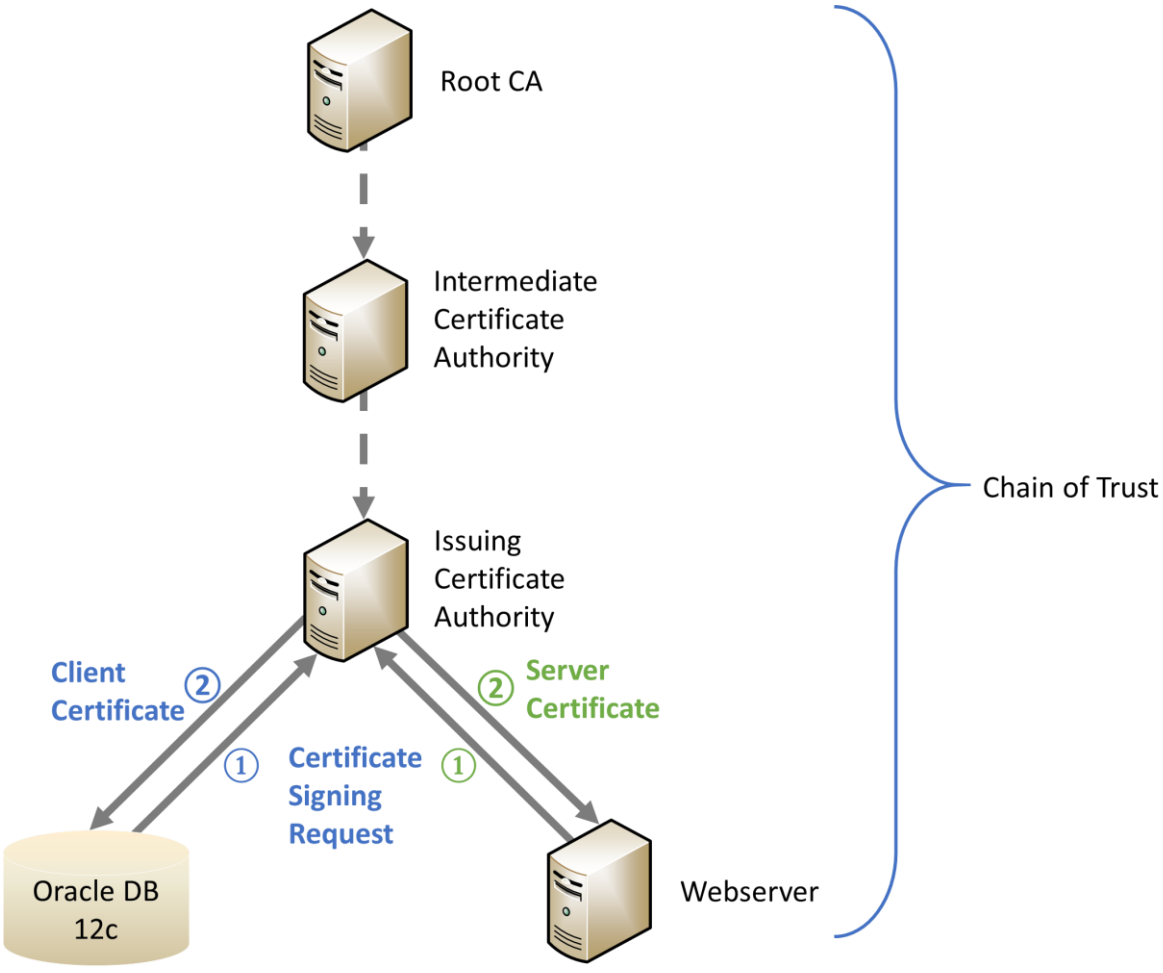

Abb. 1: PKI Aufbau und Client- & Serverzertifikat Erstellung unter Verwendung der selben issuing CA

Aufruf des Webservice

Für den Benutzer des utl_http Packages passiert der Aufruf des REST Webservices vollkommen transparent. Doch im Hintergrund greifen Verschlüsselung und Client Certificate Authentication ineinander und ermöglichen den sicheren Zugriff.

Prinzipiell erfolgen folgende Schritte ohne Zutun des Benutzers:

1. Der Client stellt einen TLSv1.2 Request, welcher die bekannten Verschlüsselungscipher-Gruppen (engl. Suites) beinhaltet, an den Webserver
2. Die Antwort des Webservers beinhaltet:
    a. Das Serverzertifikat inklusive der Zertifizierungskette
    b. Einen Certificate Request, um das Clientzertifikat zu erhalten
    c. Die gewählte Cipher-Suite
3. Der Client antwortet auf den Certificate Request mit seinem Clientzertifikat
4. Der Server validiert das Clientzertifikat mit Hilfe des öffentlichen Schlüssels der issuing CA, wählt einen verfügbaren Cipher und verifiziert, ob der Benutzer des Zertifikats autorisiert ist den Webservice zu benutzen. Falls das Clientzertifikat valide ist und der Benutzer autorisiert ist den Service zu konsumieren sendet der Server eine „`Finished`“ Antwort
5. Anschließend können Applikationsdaten (z.B. http) ausgetauscht werden

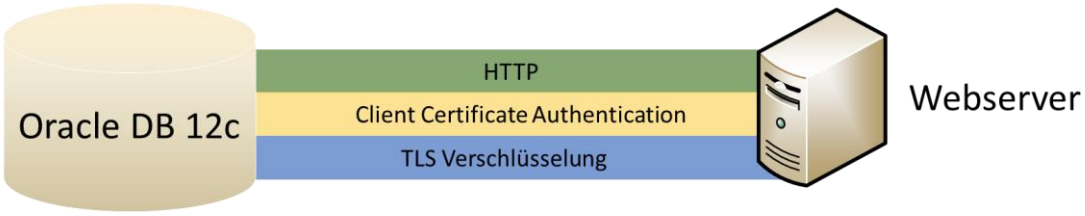

Abb. 2: PKI Aufbau und Client- & Serverzertifikat Erstellung

Um zu verifizieren, ob der Zugriff von dem Datenbank Host prinzipiell funktioniert, bietet sich auf Unix-basierten Systemen das Kommandozeilenwerkzeug curl an. So kann ausgeschlossen werden, dass sich ein Fehler bei der Erstellung des Clientzertifikats eingeschlichen hat. Eine beispielhafte Anfrage mit konfigurierten Key Store kann wie folgt aussehen:

```
curl --cert <key_store_nickname>:<password> --cacert /path/to/ca.crt
"https://depl01.vm.sphinx.at/api/v1.0/persons"
```

Auftritt Oracle
Nachdem das Clientzertifikat erfolgreich innerhalb des Oracle Wallet Managers erstellt wurde und die Zertifikatskette von der root CA bis zur issuing CA als vertrauenswürdige Zertifikate angelegt wurden, kann nun die eigentliche Arbeit in der Datenbank beginnen. Durch Access Control Entries (ACE) muss dem Datenbankbenutzer Zugriff auf das Clientzertifikat in der Wallet gegeben werden. Zusätzlich ist es notwendig eine ACE für den Webserver anzulegen und dem Datenbankbenutzer zu erlauben sich zu diesem zu verbinden und den Hostnamen aufzulösen.

Nachfolgend zwei beispielhafte ACEs:

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'depl01.vm.sphinx.at',
    ace  =>  xs$ace_type(
      privilege_list => xs$name_list('connect', 'resolve'),
      principal_name => 'SCOTT',
      principal_type => xs_acl.ptype_db)
    );
  COMMIT;
END;
/

BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_WALLET_ACE (
    wallet_path => 'file:/app/oracle/wallets/ora06/',
    ace         => xs$ace_type(
      privilege_list => xs$name_list('use_client_certificates'),
      principal_name => 'SCOTT',
      principal_type => xs_acl.ptype_db)
    );
  COMMIT;
END;
/
```

Anschließend ist es dem Benutzer scott möglich den REST Webservice mit Hilfe von utl_http abzurufen:

```
select utl_http.request
  (
    url => 'https://depl01.vm.sphinx.at/api/v1.0/persons',
    wallet_path => 'file:/app/oracle/wallets/ora06',
    wallet_password =>'doag2017'
  )
from dual;
```

Herausforderungen
Es gibt einige Stolpersteine auf dem Weg zur reibungslosen Abfrage von RESTful Werbservices unter Verwendung von Transportverschlüsselung und der Authentifizierung durch Clientzertifikate. Beispielsweise unterstützt die Oracle Database nicht alle gängigen Cipher der Transport Layer Security. Unter 12c Release 1 ist der Abruf von Webservices teilweise nicht möglich, da ein Fehler in den Netzwerkbibliotheken zu Protokollfehlern führt. Darüber hinaus erfordert es der Wallet Manager, dass eine vollständige Zertifikatskette in der Wallet gespeichert wird. Anderenfalls verwirft die Oracle Datenbank die übertragene Zertifikatskette des Servers mit dem Grund, dass diese falsch sei.

Fazit
Der sichere Konsum von RESTful Webservices mit utl_http ist mit Oracle 12c R2, unter Beachtung einiger Stolpersteine, sehr leicht umsetzbar. Gerade die Unabhängigkeit von Webservice Beschreibungen (WSDL & WADL) ermöglichen JSON Dokumente mit dem utl_http Package direkt aus der Datenbank komfortabel zu konsumieren. Allerdings bringt diese Freiheit die Abhängigkeit mit sich, dass das Dokument strikten Regeln folgt und diese dokumentiert sind. Denn falls sich der Aufbau oder Attribute aus dem Dokument ändern kann es bei der Abfrage durch SQL dazu kommen, dass sie gestern noch Ergebnisse lieferte und heute Fehlermeldungen. Hier dienen die klassischen meta-Modelle in Form der Webservice Description Languages als Versicherung.

Die Einschränkung der Datenbank auf spezifische Verschlüsselungscipher-Suites hinterlässt allerdings einen faden Beigeschmack. Immer wieder ist in den Support Portal von Oracle von Fehlern mit angeblich unterstützten Verschlüsselungsciphern-Suites zu lesen. Teilweise werden generische Fehlermeldungen für spezielle Probleme ausgegeben, was dazu führt, dass die Unterstützung des Oracle Development Team von Nöten ist. Der Weg zur sicheren Abfrage von Webservice ist bereits oft beschritten worden. Setzt man allerdings seinen Fuß ein wenig außerhalb des Weges trifft man auf undokumentierte Protokollfehler und Timeouts, welche unter früheren Releases problemlos funktionierten.

Referenzen
RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2
The Open-Source PKI Book
Oracle Database PL/SQL Packages and Types Reference
Oracle Database JSON Developer's Guide
Bug 24848928: UTL_HTTP FAILS WITH ORA-29263 IN 12.1.0.2
Bug 26862508: UTL_HTTP FAILS WITH ORA-29276 IN 12.2.0.1 SE2

Kontaktadresse:
Nico Henglmüller
Sphinx IT Consulting GmbH
Aspernbrückengasse 2/6, Wien
AT-1020 Wien

Telefon: +49 (0) 1-599 31-0
Fax: +49 (0) 1-599 31-99
E-Mail nhenglmueller@sphinx.at
Internet: www.sphinx.at