Oracle Hidden Secrets: Netzwerk-Verschlüsselung ohne jeden Aufwand

Heinz-Wilhelm Fabry, ORACLE Deutschland B.V. & Co. KG

Die Konfiguration der Verschlüsselung von Daten aus Oracle-Datenbanken bei der Übertragung über Netzwerk-Verbindungen ist ein Kinderspiel. Bis Mitte des Jahres 2013 war die Netzwerk-Verschlüsselung allerdings Teil der lizenz- und kostenpflichtigen Advanced-Security-Option (ASO) und wurde deshalb nicht von allen Anwendern genutzt. Inzwischen ist das Sicherheits-Feature aber kostenlos. Damit sind alle DBAs aufgefordert, die bisher keine Netzwerk-Verschlüsselung einsetzen, diese umgehend zu implementieren. Der Artikel beschreibt, wie das am einfachsten und in wenigen Minuten geht.

Mit Freigabe der Datenbank 12*c*, Release 1, wurde Ende Juni 2013 eine ganze Reihe von Sicherheits-Features, die bis dahin Bestandteil der Advanced-Security-Option waren, als Feature der Datenbank verfügbar – und zwar sowohl in der Enterprise Edition als auch in der Standard Edition sowie auch in den vorangegangenen Releases (wie 11*g*), sofern diese das technisch unterstützen. Damit sind die starken Authentifizierungsmethoden Kerberos, PKI und RADIUS sowie die Verschlüsselung des Netzwerks über SQL*Net in nativer Form und auch über SSL/TLS ohne zusätzliche Kosten verfügbar. Die Änderung der Lizenzbedingungen ist unter anderem unter „http://docs.oracle.com/cd/E16655_01/license.121/e17614/options.htm#DBLIC143“ im Handbuch „Oracle Database Licensing Information“ dokumentiert. Da sich die Netzwerk-Verschlüsselung über SQL*Net innerhalb weniger Minuten aktivieren lässt, wird hier die Konfiguration dieser Verschlüsselungsvariante dargestellt.

Das gesamte Netzwerk oder nur Datenbank-Zugriffe verschlüsseln?

Die Verschlüsselung des gesamten Netzwerks implementiert eine Verschlüsselung des Datenstroms zwischen den Netzwerkkarten. Das verhindert, dass durch Zugriff auf Datenleitungen Informationen in falsche Hände geraten. Sobald die Daten jedoch von einer Netzwerkkarte innerhalb eines Rechners an eine Applikation – also zum Beispiel an eine Datenbank – weitergeleitet werden, sind diese Daten nicht mehr verschlüsselt. Jeder, der Zugriff auf den Rechner hat, weil er etwa als Storage- oder System-Administrator arbeitet oder weil er sich in das Betriebssystem eingehackt hat, kann alle über das Netzwerk gesendeten Daten lesen.

Die Verschlüsselung des Netzwerks über SQL*Net verhält sich dagegen anders. Hier endet die Verschlüsselung nicht mit den Netzwerkkarten, sondern erst in den beteiligten Oracle-Komponenten: Man muss sich also in der Datenbank oder Datenbank-Anwendung einloggen und zugriffsberechtigt sein, um Daten lesen zu können. Die Verschlüsselung des Datenstroms über SQL*Net bietet daher einen höheren Schutz als die Verschlüsselung des gesamten Netzwerks.

Natürlich kommt es auf die eigene Risikoeinschätzung an, welche Form der Netzwerk-Verschlüsselung eingesetzt wird. Allerdings muss man heutzutage davon ausgehen, dass die kriminelle Energie von Individuen,

organisiertem Verbrechen und unterschiedlicher Geheimdienste irgendeine Form der Netzwerk-Verschlüsselung zwingend notwendig macht.

Verschlüsselung implementieren

Alle zurzeit durch den Oracle-Support unterstützten Datenbank- und Client-Versionen übertragen Authentifizierungs-Informationen grundsätzlich verschlüsselt. Der Schlüssel wird dabei für jede Benutzersitzung/Session immer wieder neu erstellt. Dieser Aspekt der Sicherheit ist also abgedeckt. Nach dem Aufbau der Verbindung erfolgt die gesamte Kommunikation zwischen Datenbank und Anwendung oder Remote-Datenbank allerdings prinzipiell unverschlüsselt.

Die Verschlüsselung, die implementiert werden soll, macht keinen Unterschied zwischen SQL-Befehlen und Aufrufen prozeduraler Komponenten oder Daten, die von einem Client an die Datenbank – zum Beispiel bei einem „INSERT“ und „UPDATE“ – oder von der Datenbank an einen Client, etwa als Ergebnis eines „SELECT“, geschickt werden. Der gesamte Informationsstrom ist verschlüsselt.

Im einfachsten Fall erfolgt dies über einen einzigen Eintrag in der Datei „SQLNET.ORA“, und zwar über eine Angabe zu den Parametern „SQLNET.ENCRYPTION_CLIENT“ (auch eine Datenbank kann Client sein) und/oder „SQLNET.ENCRYPTION_SERVER“. Die Einstellung „SQLNET.ENCRYPTION_SERVER=REQUIRED“ auf der Datenbank-Seite führt dazu, dass ausschließlich eine verschlüsselte Kommunikation mit dieser Datenbank erlaubt ist. Jeder andere Kommunikationsversuch wird mit einer Fehlermeldung abgelehnt.

Das ganze System ist allerdings sehr flexibel, denn neben „REQUIRED“ (Verschlüsselung ist obligatorisch) stehen für beide Seiten (Server und Client) drei weitere Einstellungen zur Verfügung:

- *REJECTED*
  Verschlüsselung wird grundsätzlich abgelehnt
- *ACCEPTED (Default)*
  Verschlüsselung wird akzeptiert, wenn der Kommunikationspartner das möchte
- *REQUESTED*
  Verschlüsselung wird gewünscht, aber nicht verlangt

Steht der Parameter für eine Datenbank beispielsweise auf „REQUESTED“, könnten die Einstellungen für die darauf zugreifenden Clients nach Wunsch unterschiedlich gesetzt sein. Während einzelne Clients über ihre „SQL*NET.ORA“-Dateien eine Verschlüsselung erzwingen („REQUIRED“), könnten andere – sofern das Sinn ergibt – darauf verzichten („REJECTED“).

Entsprechend den Einstellungen auf beiden Seiten ergibt sich somit, ob verschlüsselt wird oder nicht, beziehungsweise ob überhaupt eine Datenbank-Verbindung zustande kommt. Per Default gilt für beide Seiten der Wert „ACCEPTED“. Das hat zur Folge, dass beide Seiten für eine Verschlüsselung zur Verfügung stehen, sie diese aber nicht ausdrücklich wünschen. In der Praxis führt das dann dazu, dass keine Verschlüsselung stattfindet. Die Abbildung zeigt die möglichen Kombinationen und ihre Auswirkungen auf die Verschlüsselung beziehungsweise den Verbindungsaufbau zwischen den Kommunikationspartnern.

Neben der grundsätzlichen Festlegung, ob verschlüsselt wird oder nicht, können weitere Einstellungen sinnvoll sein. Zunächst lässt sich der Algorithmus der Verschlüsselung über die Parameter „SQLNET.ENCRYPTION_TYPES_SERVER/_CLIENT“ bestimmen. Je nach Version der Datenbank beziehungsweise des Oracle-Client stehen verschiedene Algorithmen zur Verfügung. Für ältere Datenbank-Versionen sind die verfügbaren Algorithmen im Handbuch „Advanced Security Administrators Guide“ beschrieben. Für die Datenbank 12*c* erläutert das Handbuch „Oracle Database Security Guide“ die Algorithmen. Ist kein Algorithmus festgelegt, vereinbaren Client und Server bei der ersten Kontaktaufnahme („handshake“) diesen Algorithmus zufällig aus der Reihe der gemeinsam zur Verfügung stehenden Algorithmen. Weil die Algorithmen unterschiedlich sicher und unterschiedlich performant sind, empfiehlt sich eine explizite Angabe. Der Algorithmus der Wahl ist hier „AES128“.

Darüber hinaus sind Prüfsummen-Verfahren möglich, um sicherzustellen, dass Daten im Transfer nicht modifiziert werden. Auch dies wird über Parameter in der Datei „SQLNET.ORA“ konfiguriert. Diese könnte schließlich verschiedenste Einträge enthalten (siehe Listing).

Die „CLIENT“-Parameter werden angegeben, da eine Datenbank, wie bereits mehrfach erwähnt, auch als Client agieren kann. Die gleiche Datei könnte auch auf der Client-Seite zum Einsatz kommen, vorausgesetzt der Client unterstützt die angegebenen Algorithmen. Dass auch datenbankseitige Parameter spezifiziert sind, stört dabei nicht.

Hinweise

Die Wirksamkeit der Netzwerk-Verschlüsselung lässt sich ohne fremde Hilfsmittel überprüfen. Dazu wird in der Datei „SQLNET.ORA“ nur das Tracing aktiviert „trace_level_client = 16“.

Native Verschlüsselung einschalten

Server \ Client	REJECTED	ACCEPTED	REQUESTED	REQUIRED
REJECTED	aus	aus	aus	Keine V.*
ACCEPTED	aus	aus**	ein	ein
REQUESTED	aus	ein	ein	ein
REQUIRED	Keine V.*	ein	ein	ein

* Keine Verbindung ** Default ist Accepted

Abbildung: Einstellungen von „SQLNET.ENCRYPTION_CLIENT/_SERVER“

Jede neue Datenbank-Sitzung wird ohne Neustart der Datenbank ab sofort eine Trace-Datei erzeugen. In dieser kann man dann bei nicht eingeschalteter Verschlüsselung Daten und Befehle im Klartext finden, während bei eingeschalteter Verschlüsselung nichts Sinnvolles mehr zu erkennen ist.

Die hier vorgestellte Vorgehensweise gilt für alle Client-Server-Verbindungen, die über OCI oder über Java „thick"- und „thin client"-Anwendungen arbeiten. Mögliche Ausnahmen könnten lediglich ältere „thin client"-Anwendungen betreffen. Dies muss man im Zweifelsfall testen.

Testen sollte man zur eigenen Beruhigung auch, ob das Einschalten der Netzwerk-Verschlüsselung Auswirkungen auf die Performance des Systems hat. Dies ist sehr unwahrscheinlich – jedenfalls berichten Kunden in den Veranstaltungen der Oracle Business Unit Database immer wieder, dass sie keinerlei Performance-Unterschiede zwischen einem System erkennen, das verschlüsselt, und einem, das unverschlüsselt überträgt.

```
sqlnet.encryption_server              = required
sqlnet.encryption_types_server        = AES128
sqlnet.encryption_client              = required
sqlnet.encryption_types_client        = AES128
sqlnet.crypto_checksum_server         = required
sqlnet.crypto_checksum _types_server = SHA1
sqlnet.crypto_checksum_client         = required
sqlnet.crypto_checksum _types_client = SHA512
```

Listing

Fazit:
Netzwerk-Verschlüsselung mit Oracle-Datenbankmitteln ist leicht zu implementieren und kostet weder Geld noch Performance. Diese Möglichkeiten sollte man zu seiner eigenen Sicherheit nutzen.

Heinz-Wilhelm Fabry
heinz-wilhelm.fabry@oracle.com

Tipps und Tricks aus Gerds Fundgrube

Heute: Zeituntersuchungen im Mikrosekundenbereich

Autor: Gerd Volberg, OPITZ CONSULTING GmbH

Die normalen Date-Funktionen reichen nicht aus, um Zeiträume unter einer Sekunde zu messen. Solche Untersuchungen ermöglicht der Datentyp „Systimestamp".

Das Beispiel in Abbildung zeigt, wie man diese Technik einsetzt. In der Routine „Zeitmessung" wird die Zeit untersucht, die ein Test-Loop benötigt. Dazu speichert man zwei Zeiten, deren Delta die Dauer des Loops ergibt (siehe Listing).

Im Beispiel sieht man nun, dass der Loop genau 94,688 ms dauerte. Diese Genauigkeit ist nur mit „Systimestamp" möglich.

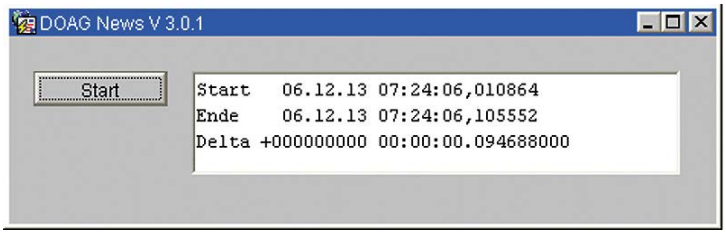

Abbildung: Das Beispiel