A business or organization can only be successful if its critical data is well managed and secure. Every day the news is filled with stories of corporations, financial institutions, and governments whose data systems have been compromised. Tightly configured network architectures which limit the direction and channels through which data flows can greatly reduce the overall exposure of application databases. Unfortunately, in my experience this aspect of data system planning is often overlooked or its importance severely underestimated. Using Oracle Connection Manager to proxy server-to-server communications and reverse proxy client-server connections adds an extra layer of security to the database’s network architecture. When properly configured and used in conjunction with firewalls and other tools like Oracle’s Advanced Security Option, it allows me as a database administrator to enforce strong configuration controls on database communications and to monitor both inbound and outbound communications for any abnormalities. In this paper I will describe my basic techniques, developed over the last several years, for adding Connection Manager to the Oracle Net network and configuring it for proxy and reverse proxy operations.
Introduction
A business or organization can only be successful if its critical data is well managed and secure. Every day the news is filled with stories of corporations, financial institutions, and governments whose data systems have been compromised. Tightly configured network architectures which limit the direction and channels through which data flows can greatly reduce the overall exposure of application databases. Unfortunately, in my experience this aspect of data system planning is often overlooked or its importance severely underestimated.

Using Oracle Connection Manager to proxy server-to-server communications and reverse proxy client-server connections adds an extra layer of security to the database’s network architecture. When properly configured and used in conjunction with firewalls and other tools like Oracle’s Advanced Security Option, it allows me as a database administrator to enforce strong configuration controls on database communications and to monitor both inbound and outbound communications for any abnormalities. (1 pp. 2-8) In this paper I will describe my basic techniques, developed over the last several years, for adding Connection Manager to the Oracle Net network and configuring it for proxy and reverse proxy operations. The techniques I describe were developed originally for use with Oracle 10gR2, but they apply equally well to the latest versions of Oracle 11g.

Network Architecture
Figure 1 illustrates a basic 3-tier application configuration with a web-based application server and Oracle database server. The basic managed network architecture I will describe in this paper utilizes two TCP/IP networks: “Public” and “Internal”. The public network includes the web client, firewalls, and application server. The internal network is non-routable and exists only between the connection manager and database server. I recommend that most actual database services – like the database listener and administrative applications like Oracle Grid Control – should be configured to run on the internal network where they cannot be subjected to random port scans, network eavesdropping or other direct attacks by remote hacker. This configuration limits the number and type of system services that have to be exposed to the public network, which is a critical concern in any hardening process.

In my sample configuration, Oracle Net traffic from the application server is reverse proxied from the public network to the internal network by Oracle Connection Manager. Likewise, outbound traffic from the database server to other servers and applications is also proxied by the Connection Manager. This allows me to tightly control and document legitimate network connections at each layer of the architecture. It also consolidates my Oracle Net connection logs and allows simpler monitoring of connection trends and errors.
As shown in the diagram, there are five critical layers I consider when deploying Oracle Connection Manager into the network architecture, each with its own configuration requirements. They are the database server, connection manager, application server, firewall, and remote Oracle Net client. As part of the deployment, it is important that each layer be hardened to limit access from higher layers in order to limit exposure of the database to direct attack. Part of that hardening process is securing the various hardware and operating systems involved. Industry standard security practices such as those recommended by the Center for Internet Security (www.cisecurity.net) or the SANS Institute (www.sans.org) should be enforced for both the network and server operating systems, as well as application clients. General database hardening steps such as those given in the Oracle Database Security Guide should also be performed. Assuming that these standards have been applied in my sample network, I can begin configuring my Connection Manager proxies.

**Database Server Configuration**

The innermost layer of my managed database network architecture is the database server itself. Within this layer I focus on three areas of interest in regard to network security: a set of initialization parameters for the database, the configuration of its local network listener, and the installation of Oracle Advanced Security Option.

**Initialization Parameters**

Oracle’s initialization parameters can affect many aspects of the Database’s behavior and overall security. There are three basic parameters I configure that affect the manner in which the database presents itself to the network: SERVICE_NAMES, LOCAL_LISTENER, and REMOTE_LISTENER.
The SERVICE_NAMES parameter determines which service names will be presented to network listeners for public connections. Service names are logical connection identifiers, independent of the Oracle System ID (SID) of the database instance. This is important because many database implementations simply use the Oracle SID as the default service name. (2 pp. 10-9 to 10-10) This default configuration would make it very difficult to restrict which applications or users are allowed to connect to a database as long as the database and its listener are running.

Whether multiple applications are sharing a particular database instance or not, each application should have its own unique service name apart from the Oracle SID (e.g. sales.mydemo.net, hr.mydemo.net, etc.). Access to each application can then be controlled, monitored or restricted independently through Oracle Connection Manager, even when the database instance is running. (2 pp. 3-3) Each service name is registered automatically by Oracle with all database listeners or connection managers identified in the LOCAL_LISTENER and REMOTE_LISTENER parameters. (2 pp. 10-9)

With Oracle 11gR2, the SERVICE_NAMES parameter should not be set directly using an ‘ALTER SYSTEM’ command in SQL*Plus. Rather, use the srvctl utility to add, update, start, stop, and remove services.

```
# srvctl add service -d DBS01 -s srv3.mydemo.net -r DBS01
# srvctl start service -d DBS01 -s srv3.mydemo.net
# srvctl status service -d DBS01 -s srv3.mydemo.net
Service srv3.mydemo.net is running
# sqlplus /nolog
SQL*Plus: Release 11.2.0.2.0 Production on Tue Jan 25 11:33:29 2011
Copyright (c) 1982, 2010, Oracle. All rights reserved.
SQL> conn / as sysdba
Connected.
SQL> show parameter service_names;
NAME                      TYPE
------------------------  ------------------------
VALUE
------------------------
service_names            string
srv1.mydemo.net, srv2.mydemo.net, srv3.mydemo.net
```

The LOCAL_LISTENER parameter specifies a name that resolves through the tnsnames.ora file (or other address repository such as Oracle Internet Directory) to an Oracle Net local listener or listeners running on the same server as the database. (2 pp. 10-10) Each listener specified will accept connections for the various service names for the database instance as specified in the SERVICE_NAMES parameter.

```
LOCAL_LISTENER = LISTENER_INTERNAL
```

A tnsnames.ora entry for a local listener might appear like the following:

```
LISTENER_INTERNAL =
_ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.0.3)(PORT = 1521))
```
The REMOTE_LISTENER parameter is similar to the LOCAL_LISTENER, except that it names a listener or listeners running on a remote server. (2 pp. 10-11) It establishes a trust relationship between the connection manager and the database server. Any listener identified in this manner will accept client connections for the database just as though it were running on the same server. Oracle Real Application Clusters use this parameter to allow cluster nodes to load balance incoming connections.

REMOTE_LISTENER can also be useful when working with Oracle Connection Manager. When the connection manager is listed as a remote listener, the database client only needs to identify the connection manager on the network. The connection manager becomes a true reverse proxy for Oracle Net communications. The actual address of the database server is completely hidden from the end user and could be changed if necessary, without having to reconfigure every remote client. This configuration may make sense in some implementations where the back end database changes frequently or if there is a need to shield users completely from the database server’s network location. However, where the security profile of the connection manager needs to be completely independent from that of the database server it should not be listed as a remote listener.

Other important parameters related to network security include OS_AUTHENT_PREFIX, OS_ROLES, REMOTE_OS_AUTHENT, and REMOTE_OS_ROLES. Industry standard best practices suggest that the OS_AUTHENT_PREFIX parameter should be set to a value other than the well known default of “ops$”. The remaining parameters should all be set to “FALSE”. (3 pp. 16-17) These values will prevent the database’s internal security from being overridden by the operating system on either the local server or a remote server.

Oracle Listener Configuration

The Oracle Listener is the last line of network defense for the database server, and I configure it to be the most restrictive line of all. By default the listener is configured to allow all incoming connections, on whatever IP addresses the server has available. As configured during database installation using Oracle’s dbca utility, the default listener.ora listener configuration file looks something like the following:

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbs01.mydemo.net)(PORT = 1521))
    )
  )

```

Once started, a quick check of the port 1521 reveals that the service is listening on all addresses, as indicated by the “*:1521” in the fourth column:

```
$ netstat -a | grep 1521
  tcp   0      0  *:1521    *:* LISTEN

```

In a well managed configuration, I want to ensure that the listener is only accepting connections from the internal, non-routable network. This minimizes the number and type of services that are exposed to
Deploying an Oracle Connection Manager

by Peter Magee

the public, and thus to direct attack from hackers. Reconfiguring the HOST parameter in the listener.ora file will restrict the listener to accepting connections only on a particular IP address. In this case, the address of the interface on the internal network¹:

```
LISTENER_INTERNAL =
   (DESCRIPTION_LIST =
       (DESCRIPTION =
           (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.0.3)(PORT = 1521))
        )
   )
```

A recheck of the port after the listener is restarted shows that the service is now only available on the address I want.

```
$ netstat -a | grep 1521
tcp   0   0 dbs01-int.demo.com:1521      *:*       LISTEN
```

Additionally, I believe it is a good idea to restrict the clients from which the database server will accept incoming connections. Since all legitimate inbound connections should come through the connection manager, which will filter client connections using its own rules, there is a very short list of acceptable clients which should connect directly to the database server. I use Oracle Valid Node checking to limit the acceptable client list by adding a few lines to the sqlnet.ora file, as follows:

```
# Configure TNS firewall to loopback and local IP address only
TCP.VALIDNODE_CHECKING = YES
TCP.EXCLUDED_NODES = (*..*.*.*)
TCP.INVITED_NODES = (127.0.0.1, 192.168.0.2, 192.168.0.3)
```

¹ It is important to note that there is a change in default listener behavior when using Oracle Grid Infrastructure 11.2.0.2. The host specification in the listener.ora file is ignored in favor of a high availability service parameter which continues to force the listener to attach to all network interfaces on the server. This parameter must be disabled for the host restriction in defined listener.ora to take effect.

```
# crsctl stat res ora.LISTENER.lsnr -p
NAME=ora.LISTENER.lsnr
TYPE=ora.listener.type
...
ENDPOINTS=TCP:1521
...
# crsctl modify resource ora.LISTENER.lsnr -attr "ENDPOINTS="
# srvctl stop listener -l LISTENER
# srvctl start listener -l LISTENER
# crsctl stat res ora.LISTENER.lsnr -p
NAME=ora.LISTENER.lsnr
TYPE=ora.listener.type
...
ENDPOINTS=
...
The TCP_INVITED_NODES parameter is an explicit list of all IP addresses which are allowed to connect to
the listener. (2 pp. 9-4) The settings shown will allow inbound connections from the localhost and the
connection manager, and reject connections from anywhere else.

Industry standard best practices for hardening the listener call for additional configuration settings in
the listener.ora and sqlnet.ora files on the server. (1 pp. 2-7) I always set a listener administration
password using the Lsnrctl utility, and set the ADMIN_RESTRICTIONS_[listener-name] parameter in the
listener.ora file to ON. This prevents a hacker from reading the listener configuration remotely or from
submitting “SET” commands to the listener that would alter its parameters. (4 pp. 7-9) To prevent
unauthorized client versions and terminate bad or hung connection attempts, I set the
SQLNET.ALLOWED_LOGIN_VERSION\(^2\) and the SQLNET.INBOUND_CONNECT_TIMEOUT parameters in the
sqlnet.ora file to values appropriate for whatever organization I am working for. I also remove the
external procedure configuration whenever possible.

A final check of the listener status should show the appropriate network and security parameters in use.
Note that each service name status is represented, in addition to the Oracle SID.

```
$ lsnrctl status listener
LSNRCTL for Linux: Version 11.2.0.2.0 - Production on 25-JAN-2011 09:53:06
Copyright (c) 1991, 2010, Oracle. All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.0.3)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias LISTENER
Version TNSLSNR for Linux: Version 10.2.0.4.0 - Production
Start Date 25-JAN-2011 10:35:41
Uptime 35 days 0 hr. 17 min. 25 sec
Trace Level off
Security ON: Password or Local OS Authentication
SNMP ON
Listener Parameter File /opt/oracle/db/network/admin/listener.ora
Listener Log File /opt/oracle/db/network/log/oratns.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=192.168.0.3)(PORT=1521))
Services Summary...
Service "srvt01.mydemo.net" has 1 instance(s).
  Instance "DBS01", status READY, has 1 handler(s) for this service...
Service "srvt02.mydemo.net" has 1 instance(s).
  Instance "DBS01", status READY, has 1 handler(s) for this service...
Service "srvt03.mydemo.net" has 1 instance(s).
  Instance "DBS01", status READY, has 1 handler(s) for this service...
Service "DBS01.mydemo.net" has 1 instance(s).
  Instance "DBS01", status READY, has 1 handler(s) for this service...
Service "DBS01_XPT.mydemo.net" has 1 instance(s).
  Instance "DBS01", status READY, has 1 handler(s) for this service...
The command completed successfully
```

\(^2\) The ALLOWED_LOGIN_VERSION setting may block connections for some JDBC clients.
Oracle Advanced Security

Oracle Advanced Security (OAS) is a separately licensed option for Oracle Database Server. While that does imply some extra cost, OAS provides strong user authentication and allows all database network traffic to be encrypted. (5 pp. 1-3) I believe that these features are essential in a fully managed and secured environment. Indeed, in many industrial, medical, or military applications it may be required. OAS becomes the glue that holds the rest of the architecture together because it enables the various layers to communicate without fear of eavesdropping or data stream alteration.

By default, username and password are always encrypted when a connection is originally established, but subsequent communications are not encrypted unless OAS is configured. (6) With OAS in place, it is relatively simple to implement extremely strong encryption, including newer AES algorithms. I recommend that OAS be installed and configured on the database server, connection manager, and Oracle Net client to require the highest level of encryption possible. A basic, Oracle Net client configuration that requires strong encryption would include the following parameters in the sqlnet.ora file: (5 pp. A-2 to A-7)

```sql
# Settings for when this client is connecting to a server.
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT= (SHA1)
SQLNET.CRYPTO_CHECKSUM_CLIENT = required
SQLNET.ENCRYPTION_TYPES_CLIENT= (AES256)
SQLNET.ENCRYPTION_CLIENT = required

# Settings for when this client is connecting to a server.
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT= (SHA1)
SQLNET.CRYPTO_CHECKSUM_CLIENT = required
SQLNET.ENCRYPTION_TYPES_CLIENT= (AES256)
SQLNET.ENCRYPTION_CLIENT = required

# Settings for when this client is connecting to a server.
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT= (SHA1)
SQLNET.CRYPTO_CHECKSUM_CLIENT = required
SQLNET.ENCRYPTION_TYPES_CLIENT= (AES256)
SQLNET.ENCRYPTION_CLIENT = required

# Seed needs to be randomly generated consisting of between
# 10 and 70 characters. This seed should be different for each host.
SQLNET.CRYPTO_SEED = somerandomalphanumericstringofabout70characters
```

A similar configuration is required in the sqlnet.ora file on the database server. Note that the server configuration also includes client values for use in database links.

```sql
# Settings for when a client is connecting to this server.
# Incoming connections to database must be checksum'd and encrypted.
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER= (SHA1)
SQLNET.CRYPTO_CHECKSUM_SERVER = required
SQLNET.ENCRYPTION_TYPES_SERVER= (AES256)
SQLNET.ENCRYPTION_SERVER = required

# Settings for when this client is connecting to a server.
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT= (SHA1)
SQLNET.CRYPTO_CHECKSUM_CLIENT = required
SQLNET.ENCRYPTION_TYPES_CLIENT= (AES256)
SQLNET.ENCRYPTION_CLIENT = required

# Seed needs to be randomly generated consisting of between
# 10 and 70 characters. This seed should be different for each host.
SQLNET.CRYPTO_SEED = somerandomalphanumericstringofabout70characters
```

These settings require the Oracle Net client and the database server to encrypt their communication using the 256-bit AES algorithm, with SHA-1 check-summing. If the encryption cannot be established, then the connection will be rejected. For more Oracle Advanced Security configuration options,
including strong user authentication and encryption using PKI certificates, read the Oracle Database Advanced Security Administrator Guide.

**Oracle Connection Manager Configuration**
The second layer of my managed database network architecture is the connection manager. In Oracle 11gR2, Connection Manager can be installed from the same media as the Oracle Client – it is included in the Enterprise Edition license and is available as an option under the “Custom” install\(^3\). Similar to the Oracle Listener, its system resource requirements are very low. It can be installed on separate hardware from the database server, or even on the same host. I recommend separate hardware whenever possible, however, as this allows outbound communications from the database server to be proxied as well as the inbound connections. If the same hardware must be used, then install Connection Manager under a separate user account with a separate Oracle Home directory from other Oracle software.

Oracle Connection Manager allows rule-based, firewall-like filtering of incoming Oracle Net connections to the database server. For example, connections from specific client systems can be limited to connecting with particular database services on particular hosts. (2 pp. 1-14) This allows me to very narrowly define which are legitimate connections and greatly increases the level of protection around the database server.

Connection Manager’s cman.ora configuration file contains three sections: the basic configuration information about the network address and port number, a rules list that defines legal and illegal connections, and a parameter list that allows the DBA to set other configuration options. (4 pp. 8-1) A portion of the connection manager’s cman.ora configuration file is shown below. Similar to the default listener configuration, a basic connection manager configuration will listen for database connection requests on port 1521 of all IP addresses available on the system. I recommend this configuration on the connection manager because it allows the administrator to filter both inbound and outbound connections. The empty host parameter allows the connection manager to listen to port 1521 on all available IP addresses.

```sql
C:
configuration=  
 (address=(protocol=tcp)(host=)(port=1521))  
(rule_list=  
 ...  
)  
(parameter_list=  
 ...  
)
```

---

\(^3\) Prior to 11gR2, Connection Manager was available in the database installation media under the “Advanced” install option. The minimum required options are Oracle Call Interface, Oracle Net, Oracle Connection Manager, and Oracle Net Listener.
A connection rule list might look something like the following:

```
(rule_list=
  # INBOUND RULES
  # - Application Server 1
  (rule=(src=172.16.0.1)(dst=192.168.0.3)(srv=svc1.mydemo.net)(act=accept))
  # - DBA workstations
  (rule=(src=172.16.1.0/24)(dst=*)(srv=*)(act=accept))
  #
  # OUTBOUND RULES
  # - Remote DB Server
  (rule=(src=192.168.0.3)(dst=172.16.1.3)(srv=remote.mydemo.net)(act=accept))
  #
  # Local Connections
  (rule=(src=172.16.0.3)(dst=127.0.0.1)(srv=*)(act=accept))
  (rule=(src=cnx01)(dst=127.0.0.1)(srv=cmon)(act=accept))
  (rule=(src=cnx01-int)(dst=127.0.0.1)(srv=cmon)(act=accept))
  #
  # All other source IPs
  (rule=(src=*)(dst=*)(srv=*)(act=drop))
}
```

Note in this example that the application server is only allowed to connect to the service svc1 on the database server, not svc2 or svc3. DBA workstations are allowed to connect from anywhere on the 172.16.1.0 network to any database service name. One outbound connection is permitted from 192.168.0.3 to a remote server at 172.16.1.3. All other inbound or outbound connections are dropped, generating a TNS-12537 error on the client and a TNS-12529 or TNS-01186 return code in the connection manager log. Having connection rules be so granularly defined makes configuration control and network monitoring simpler. It allows both inbound and outbound connections to be logged in detail, whether they are successful or – in particular – if they are not. If a connection is made (or attempted) that bypasses the Connection Manager, it can immediately be identified as suspicious.

The parameter list section of cman.ora contains several important settings as well. The INBOUND_ and OUTBOUND_CONNECTION_TIMEOUT parameters set timeout limits for incoming and outgoing connections to be established. The LOG_LEVEL parameter determines the amount of detail to be included in the Connection Manager’s log file. The REMOTE_ADMIN parameter allows or disallows remote administrative access to the connection manager. (4 pp. 8-6 to 8-8) It is also important to set an administration password using the cmctl utility, similar to the administration password for the network listener. Additional parameters are defined in the Oracle Database Net Services Reference.

```
(parameter_list=  
  (inbound_connection_timeout=60)
  (outbound_connection_timeout=60)
  (log_level=admin)
  (remote_admin=no)
  )
```
**Logging**

Both Oracle Connection Manager and Network Listener produce log files with a great deal of useful information. Monitoring these logs is an essential part of maintaining the managed database network. A sample log snippet is shown here:

```
25-JAN-2011 15:32:12 * service_update * cman * 0
25-JAN-2011 15:33:14 * (CONNECT_DATA=(SERVICE_NAME=svc1.mydemo.net))
   (CID=(PROGRAM=C:\WINDOWS\system32\dllhost.exe)(HOST=APP01)(USER=srvcapp)) *
   (ADDRESS=(PROTOCOL=tcp)(HOST=192.168.0.1)(PORT=1358)) * establish * svc1.mydemo.net * 0
25-JAN-2011 15:33:14 * (CONNECT_DATA=(SERVICE_NAME=svc1.mydemo.net))
   (CID=(PROGRAM=C:\WINDOWS\system32\dllhost.exe)(HOST=APP01)(USER=srvcapp)) *
   (ADDRESS=(PROTOCOL=tcp)(HOST=192.168.0.1)(PORT=1359)) * establish * svc1.mydemo.net * 0
25-JAN-2011 15:33:19 * (CONNECT_DATA=(SERVICE_NAME=svc2.mydemo.net))
   (CID=(PROGRAM=C:\WINDOWS\system32\dllhost.exe)(HOST=APP01)(USER=srvcapp)) *
   (ADDRESS=(PROTOCOL=tcp)(HOST=192.168.0.1)(PORT=1361)) * establish * svc2.mydemo.net * 12529
```

It is possible to identify the network address and OS username of the client, its target database, and the error number (if any) of each connection. I often write shell scripts, configure Oracle Grid Control, or use a 3rd party tool like Splunk to monitor my connection manager logs for errors so that I can receive alerts or e-mails in real-time when there are persistent problems.

**Application Server & Client Configuration**

The third layer of the managed database network architecture is the application server. It is essentially the same as the fifth layer, the Oracle Net client, except for their relative positions in regards to the firewall. These are the levels at which the Oracle Client – OCI, JDBC OCI, or JDBC Thin – is installed. Each client type has its own characteristics and hardening requirements.

**OCI Client**

The Oracle Call Interface (OCI) client is the standard, “thick” client used by applications that need the full Application Programming Interface (API) library of Oracle commands. Among other things, the OCI client allows the use of strong encryption and authentication methods that are transparent to the application itself. It can be installed on a client workstation or an application server, and is also installed by default on the database server to enable server-to-server database links.

OCI clients resolve database network addresses using the tnsnames.ora configuration file, or some other network name repository such as Oracle Internet Directory (OID). (2 pp. 8-3) Although OID offers additional security and configuration management benefits, most clients still use the tnsnames.ora file because of its simplicity and lack of additional cost. A traditional tnsnames.ora entry for a TCP/IP network might look like the following example. The PROTOCOL, HOSTNAME, PORT, and SERVICE_NAME attributes describe a unique connection path from the client to the server. Note the absence of the

---

4 Strong encryption and authentication requires the purchase and installation of the Oracle Advanced Security Option.
common ORACLE_SID parameter, which was replaced as the default Oracle Net database identifier by SERVICE_NAME in Oracle 8i.

```sql
DBS01 =
{(DESCRIPTION =
 (ADDRESS_LIST =
  (ADDRESS = (PROTOCOL = TCP)(HOST = dbs01.mydemo.net)(PORT = 1521))
 )
 (CONNECT_DATA =
  (SERVICE_NAME = srv1.mydemo.net)
 )
)}
```

The addition of the Oracle Connection Manager to the network architecture requires some changes to be made to the default client configuration. If the connection manager is configured as a remote listener (i.e. full reverse proxy) for the database server, then the client tnsnames.ora entry should point to the connection manager hostname instead of the database server. The connection manager will translate the requested service name to a final destination based on its own connection rule set, and no other configuration of the client will be required.

```sql
DBS01 =
{(DESCRIPTION =
 (ADDRESS_LIST =
  (ADDRESS = (PROTOCOL = TCP)(HOST = cnx01.mydemo.net)(PORT = 1521))
 )
 (CONNECT_DATA =
  (SERVICE_NAME = srv1.mydemo.net)
 )
)}
```

If the connection manager is not listed as a remote listener for the database server, then the client tnsnames.ora entry needs to specify both the connection manager and the database server. In this configuration the connection manager acts as an application firewall, but doesn’t completely obfuscate the location of the database server like a reverse proxy would. Using the SOURCE_ROUTE parameter, the tnsnames.ora entry instructs the client that there is a multi-step path involved in the connection. (2 pp. 13-3)

```sql
DBS01 =
{(DESCRIPTION =
 (ADDRESS_LIST =
  (SOURCE_ROUTE = on)
  (ADDRESS = (PROTOCOL = TCP)(HOST = cnx01.mydemo.net)(PORT = 1521))
  (ADDRESS = (PROTOCOL = TCP)(HOST = dbs01-int.mydemo.net)(PORT = 1521))
 )
 (CONNECT_DATA =
  (SERVICE_NAME = srv1.mydemo.net)
 )
)}
```
Because the database server only listens for incoming connections on the Internal, non-routable network, the hostname used for the database server must be resolved by the connection manager to a server on that network. Hostnames for non-routable IP addresses should not be listed in public DNS or the local hosts file of the client or application server. Only the public name of the connection manager should be resolvable through DNS. To allow the hostname to be resolved without disclosing the architecture of the Internal network, I configure the local “hosts” file of the Connection Manager to include all local public and internal network hostnames and IP addresses that relate to Oracle Net communications. This allows clients to connect to the database server without knowing its true network address. The local hosts file on my connection manager might look something like the following example:

```bash
$ more /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1 localhost.localdomain localhost
172.16.0.1 app01.mydemo.net app01
172.16.0.2 cnx01.mydemo.net cnx01
172.16.0.3 dbs01.mydemo.net dbs01
192.168.0.2 cnx01-int.mydemo.net cnx01-int
192.168.0.3 dbs01-int.mydemo.net dbs01-int
```

### JDBC OCI Client

The JDBC OCI Client adds a JDBC interface layer to the standard Oracle OCI client. It converts JDBC calls to OCI calls before sending them over Oracle Net to the database. (2 pp. 5-5) It uses the same name resolution services and is capable of providing the same level of strong authentication and encryption using Oracle Advanced Security (OAS). Given the same tnsnames.ora entry that was provided earlier, a connection URL for the JDBC OCI Client would look like the following, where “DBS01” is the connection alias in the tnsnames.ora file.

```sql
jdbc:oracle:oci:@DBS01
```

### JDBC Thin Client

The JDBC Thin Client is commonly used by Java applets. It establishes a direct connection to the database using Java sockets. (2 pp. 5-5) It contains a stripped down, “thin” implementation of Oracle Net, and does not support name resolution using tnsnames.ora, or the same levels of authentication or encryption included in OAS. A typical, default connection URL for a JDBC Thin client contains the database server hostname, listener port number, and Oracle SID. Depending on the JDBC driver’s version, it may also allow the SERVICE_NAME parameter.

```sql
jdbc:oracle:thin:@dbs01.mydemo.net:1521:DBS01 (using ORACLE_SID)
jdbc:oracle:thin:@//dbs01.mydemo.net:1521/srv1.mydemo.net (using SERVICE_NAME)
```
Since Oracle Connection Manager requires the SERVICE_NAME to be specified, a change to the connection URL is required. There are three options. The first option I recommend is to convert the client to using the JDBC OCI client. This is accomplished by changing the “thin” option to “oci”, and replacing the hostname:port:SID string with the tnsnames.ora alias. This allows the application to inherit all the benefits of the OCI client, including OAS and Oracle RAC support. In some 3rd party applications, however, this may not be possible if the connect string format is hardcoded.

A second option, if the connection manager is configured as a remote listener for the database server and that the connection URL is configurable by the administrator, is to replace the hostname:port:SID string with the entire contents of the tnsnames.ora entry for the database. This would create a connection URL like the following:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=cnx01.mydemo.net)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=srv1.mydemo.net)))
```

The third, least desirable option is to modify the connection manager rule to allow connections to all service names. Replacing the service name in the connection rule with a wildcard star will allow the connection request to proceed with an SID instead of the service name. I try to avoid using this configuration, as it effectively removes one of the key security restrictions made available by Connection Manager. All service names on the database server would now be exposed to this client, even if access is not required.

```
# INBOUND RULES
# = Application Server 1
(rule=(src=172.16.0.1)(dst=192.168.0.3)(srv=*)(act=accept))
```

**Command Line Tools**

I believe that one other item should be mentioned when hardening application servers and other remote clients in a well managed network. Command line database tools such as SQL*Plus which allow ad-hoc SQL queries, commands, and scripts to be executed should not be installed on client systems in production environments unless absolutely necessary. Particularly on application servers, which should only access the database through designed, controlled user interfaces, and connection managers, which should not connect directly to the database themselves but only proxy network traffic from other clients. This follows the principle of least privilege, in which no user or system should have access to more privileges – including software – than absolutely necessary to complete the tasks required. (1 pp. 2-4)

**Firewall Configuration**

The fourth layer of my managed network architecture is the firewall. Firewalls provide a protective layer of filtering over the entire network, establishing which traffic is allowed to leave or enter a particular enclave, and from or to which servers. I recommend that data paths be restricted as much as possible to prevent the misdirection of data to unauthorized recipients. For example, in the 3-tier application
architecture shown in Figure 2, Firewall 1 allows outbound traffic from a user’s web client on port 443. Firewall 2 allows inbound traffic on port 443 to pass through to the application server. No Oracle Net traffic on port 1521 is allowed to enter or leave the application’s network enclave.

![Figure 2: Web Client Connection](image)

If Oracle Net traffic must pass the firewall, either to or from the database server, it should only be allowed to go to or originate from the connection manager. This prevents remote users from bypassing the connection manager’s security rules and logging, and enforces configuration management standards on production data interfaces. Figure 3 illustrates a configuration in which Firewall 1 allows outbound traffic from the user’s Oracle Net Client to proceed to Firewall 2. Firewall 2 allows inbound traffic from Firewall 1 to proceed to the connection manager. Firewall 2 also allows outbound Oracle Net traffic from the connection manager on port 1521 to proceed to Firewall 3. Firewall 3 allows the inbound Oracle Net traffic from Firewall 2 to proceed to the remote database server. All data transport paths are tightly controlled, preventing remote network scans on, or undocumented connections to or from a database server.
High Availability
A key component of many production database architectures is redundancy. The basic architecture I have described previously includes several single points of failure. There are many actions that can be taken to overcome these failure points: The database server can be clustered using Oracle RAC technology, firewalls and application servers can also be clustered and load balanced to provide greater throughput and redundancy. What about the connection manager?

Unfortunately, Oracle Connection Manager cannot yet be included as a cluster-aware RAC component, particularly if installed on separate hardware from the database servers. Multiple independent connection managers can be installed on the network, however, to provide load balancing and redundancy in the connection path between the application server or Oracle Net client and the database server. (2 pp. 5-12) I have tested several configurations, each of which supports a different type of overall architecture.

Single Database Configuration
The simplest way I have found to provide redundancy at the connection manager layer is to provide multiple, parallel connection managers, as shown in Figure 4. In this way, a set of connection managers could be configured to reverse proxy Oracle Net communications from application clients into an entire farm of stand-alone database servers.
Figure 4: Redundant Connection Managers

This configuration increases the complexity of the Oracle Net client’s tnsnames.ora file. If the connection managers are listed as remote listeners, allowing both of them to be reverse proxies, the tnsnames.ora entry would look like the following example. The LOAD_BALANCE parameter instructs the client to alternate connections between the addresses listed in a random pattern. (4 pp. 6-7) If one connection manager is down, the client will automatically cycle new connections to another one in the list.

```
DBS01 =
{(DESCRIPTION =
    {ADDRESS_LIST =
       {LOAD_BALANCE = on}
       {ADDRESS = (PROTOCOL = TCP)(HOST = cnx01.mydemo.net)(PORT = 1521))
       {ADDRESS = (PROTOCOL = TCP)(HOST = cnx02.mydemo.net)(PORT = 1521))
    }
    {CONNECT_DATA =
        {SERVICE_NAME = srv1.mydemo.net}
    })
}
```

If the connection managers are not configured as remote listeners, then the tnsnames.ora entry becomes a little more complex. Now load balancing must occur between two distinct connection descriptions instead of individual addresses, each of which includes a SOURCE_ROUTE path through a different connection manager. (7)
Oracle RAC Configuration

In an Oracle RAC environment, the configuration may become even more complex. Beginning in Oracle 11gR2 the REMOTE_LISTENER parameter is configured by default to point to the virtual SCAN IP address — that is, the virtual network address pool by which the entire cluster is identified logically on the network, as shown in Figure 5.

```
DBS01 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (SOURCE_ROUTE = on)
        (ADDRESS = (PROTOCOL = TCP)(HOST = cnx01.mydemo.net)(PORT = 1521))
        (ADDRESS = (PROTOCOL = TCP)(HOST = dbs01-int.mydemo.net)(PORT = 1521))
      )
      (CONNECT_DATA =
        (SERVICE_NAME = srv1.mydemo.net)
      )
    )
    (DESCRIPTION =
      (ADDRESS_LIST =
        (SOURCE_ROUTE = on)
        (ADDRESS = (PROTOCOL = TCP)(HOST = cnx02.mydemo.net)(PORT = 1521))
        (ADDRESS = (PROTOCOL = TCP)(HOST = dbs01-int.mydemo.net)(PORT = 1521))
      )
      (CONNECT_DATA =
        (SERVICE_NAME = srv1.mydemo.net)
      )
    )
  )
```

Figure 5: Default RAC Remote Listeners

```
REMOTE_LISTENER = (ADDRESS=(PROTOCOL=TCP)(HOST=dbs01-scan.mydemo.net)(PORT=1521))
```

To add Connection Managers to this configuration, we must define a tnsnames.ora alias that includes all of the remote listeners, and then override the default setting.
Once the tnsnames.ora alias for the remote listeners is defined, I can reset the REMOTE_LISTENER parameter in the database to point to the alias, ensuring that all cluster nodes register with each listener automatically. Then I can configure each application server to use the pool of Connection Managers instead of the SCAN IP address pool. This also allows me to keep the cluster (with the SCAN address pool) on a separate physical network from the Application Servers, using the Connection Manager pool to bridge the gap.

```
# sqlplus /nolog
SQL*Plus: Release 11.2.0.2.0 Production on Tue Jan 25 16:36:44 2011
Copyright (c) 1982, 2010, Oracle. All rights reserved.
SQL> conn / as sysdba
Connected.
SQL> alter system set remote_listener='LISTENERS_PUBLIC' scope=both;
System altered.
SQL> show parameter remote_listener

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>remote_listener</td>
<td>string</td>
<td>LISTENERS_PUBLIC</td>
</tr>
</tbody>
</table>
```

Figure 6: RAC Connection Managers
If the Connection Managers in the RAC configuration are not configured as remote listeners, then each Oracle Net client or application server must know all possible connection paths to each virtual IP address of the cluster, through each connection manager. Now the number of connection descriptions in the tnsnames.ora file will equal the number of connection managers times the number of virtual IP addresses. For a two-node cluster with two connection managers like the network shown in Figure 6 there are four possible connection paths between the application server and the database. This configuration results in a tnsnames.ora entry like the one on the next page. If the cluster contains more than a few nodes, however, the connection string may quickly grow too long for some applications to handle and become difficult to modify.

```sql
DBS01 =
 (DESCRIPTION_LIST =
 (LOAD_BALANCE = YES)
 (DESCRIPTION =
 (ADDRESS_LIST =
 (SOURCE_ROUTE = on)
 (ADDRESS = (PROTOCOL = TCP)(HOST = cnx01.mydemo.net)(PORT = 1521))
 (ADDRESS = (PROTOCOL = TCP)(HOST = cnx02.mydemo.net)(PORT = 1521))
 )
 (CONNECT_DATA =
 (SERVICE_NAME = srv1.mydemo.net)
 )
 )
 (DESCRIPTION =
 (ADDRESS_LIST =
 (SOURCE_ROUTE = on)
 (ADDRESS = (PROTOCOL = TCP)(HOST = cnx01.mydemo.net)(PORT = 1521))
 (ADDRESS = (PROTOCOL = TCP)(HOST = cnx02.mydemo.net)(PORT = 1521))
 )
 (CONNECT_DATA =
 (SERVICE_NAME = srv1.mydemo.net)
 )
 )
 (DESCRIPTION =
 (ADDRESS_LIST =
 (SOURCE_ROUTE = on)
 (ADDRESS = (PROTOCOL = TCP)(HOST = cnx01.mydemo.net)(PORT = 1521))
 (ADDRESS = (PROTOCOL = TCP)(HOST = cnx02.mydemo.net)(PORT = 1521))
 )
 (CONNECT_DATA =
 (SERVICE_NAME = srv1.mydemo.net)
 )
 )
 (DESCRIPTION =
 (ADDRESS_LIST =
 (SOURCE_ROUTE = on)
 (ADDRESS = (PROTOCOL = TCP)(HOST = cnx01.mydemo.net)(PORT = 1521))
 (ADDRESS = (PROTOCOL = TCP)(HOST = cnx02.mydemo.net)(PORT = 1521))
 )
 (CONNECT_DATA =
 (SERVICE_NAME = srv1.mydemo.net)
 )
 )
 )
 ```
In this instance, one method I have found to reduce tnsnames.ora complexity is to limit which connection managers can talk to each node. For instance, in a four-node Oracle RAC with two connection managers and clustered or load balanced application servers, I may configure the various tnsnames.ora files so that each application server uses different default paths to the database nodes, as shown in Figure 6. The size of the tnsnames.ora entry is then kept as small as possible, with each application server having four initial connection paths to the cluster. Each connection manager should contain rules that allow each application server to connect to each database server in the Oracle RAC. This will allow internal RAC load balancing to take place, maintaining full connection path redundancy and ensuring that neither application server would be completely disabled by the loss of a single connection manager.

Another method I have used to reduce complexity in a tnsnames.ora entry applies strictly to web-based applications, where the end user accesses a middle-tier application server but does not directly access the database server. In this case, application servers could be configured with an interface on the internal network, and the ValidNode list of acceptable clients on the database server expanded to include them. The application servers then become the proxy for incoming connections, using a more “traditional” RAC connection string.
Stand-alone connection managers could still be used to proxy outbound server-to-server connections, resulting in a traffic flow that looks like the following:

```sql
DBS01_VIP =
  (DESCRIPTION =
   (ADDRESS_LIST =
     (LOAD_BALANCE = on)
     (ADDRESS = (PROTOCOL = TCP)(HOST = dbs01-vip.mydemo.net)(PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)(HOST = dbs02-vip.mydemo.net)(PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)(HOST = dbs03-vip.mydemo.net)(PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)(HOST = dbs04-vip.mydemo.net)(PORT = 1521))
   )
   (CONNECT_DATA =
     (SERVICE_NAME = srv1.mydemo.net)
   )
  )

DBS01_SCAN =
  (DESCRIPTION =
   (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(HOST = dbs01-scan.mydemo.net)(PORT = 1521))
   )
   (CONNECT_DATA =
     (SERVICE_NAME = srv1.mydemo.net)
   )
  )
```

The drawback to this configuration is the removal of the Oracle Net-aware reverse proxy. It also creates separate log files for inbound and outbound communications and increases the complexity of monitoring. On the positive side, this arrangement still allows me to maintain a low public network profile and to manage and document all inbound and outbound connections. Rogue connections or connection attempts are still easily identified.
Conclusion
Securing the data of any business or organization is critical to its success. The implementation of Oracle Connection Manager in managed database network architectures allows me as an administrator to monitor and enforce configuration management standards on both inbound and outbound database communications by defining explicitly which database connections are valid. With its ability to proxy or reverse proxy Oracle Net connections based on source address, destination address, and service name, it allows me to control the paths and direction through which data moves, adding an extra layer of protection to the Oracle Net environment by obfuscating my network configuration. Using Connection Manager to physically isolate the database server’s network listener from the public network, this architecture can greatly reduce the ability of a hacker to intercept data or launch direct attacks on the database server and make such attempts easier for me to detect. Connection Manager can also play a key role in load balancing and high availability configurations, ensuring that the right applications will always have access to the organization’s critical data. When planning a well-managed, secure Oracle database network environment, Connection Manager should be a central component of every design.
Works Cited
3. The Center for Internet Security. *Center for Internet Security Benchmark for Oracle 9i/10g Ver. 2.0.*

Biography
Peter Magee is a CompTIA Security+ certified Oracle database administrator with 16 years of professional experience in both commercial and military computing environments. Over the years he has published and presented white papers and scripts on a variety of performance tuning and security subjects. Highlights of Mr. Magee’s career include contributing content and scripts to the Defense Information Systems Agency’s first Database Security Technical Implementation Guide, presentation of papers on real-time auditing and monitoring techniques using Oracle Enterprise Manager at the Independent Oracle Users Group (IOUG) Live ’99 Conference, and participation as a network defender in the Air Force’s “Black Demon” cyber defense exercise. His scripts on security and system monitoring have been published on the Oracle Magazine Interactive Web Site, the IOUG’s “Library of Oracle Knowledge”, the Sun Microsystems “Big Admin” portal, the Oracle Communities portal, and the CSC Leading Edge Forum. Mr. Magee is currently employed as a Lead Database Architect at Computer Sciences Corporation, where he provides production operations support for a customer’s 13,000 user business management system.