Database - The Final Firewall

Brian Suddeth

Database - The Final Firewall
S. Brian Suddeth
1/28/02

ABSTRACT

The "crown jewels" of most corporations lie in its data. Whether it is corporate identity, trade secrets, business practices, contractual negotiations, financial records, customer records or other privileged information, this is where the most tightly guarded assets lie. Most organizations have this information organized in a database management system. In following industry best practices for information security, all organizations must setup layers of protection called "defense in depth" to safeguard these assets. Every major layer of defense has many sub-layers, each of which is an integral part of your defense. The last layer of protection before the raw data is usually your database management system itself. The database can be used not only to store and organize your information, but it contains intrusion detection tools that can help to locate and isolate data theft or misuse. It is your final firewall. Multiple layers of security may be set in this last line of defense, helping to control access, monitor usage, set tripwires for intrusions, and attempt to maintain evidence needed if intrusions or misuse occur.

=======

America grew up as an agricultural economy. It quickly progressed to an industrial economy. Today the paradigm has shifted again, and our prosperity is now tied to the processing of information. The protection of that information is the goal of the Information Security industry. We in information security are fighting battles of information warfare on a daily basis, both directly and indirectly. The battles can be violent, but sometimes they can also be quiet and virtually unseen. It takes many techniques to locate unseen or unsuspected attacks.

Information security is implementable at many levels. As each level is analyzed, it can be expanded to provide cross checking of information, redundant logging, validation of content, and early warning systems for possible intrusions. Many papers have been written covering multiple aspects of firewall configuration, network intrusion detection systems (NIDs), virus detection systems, etc, but few have taken a practical look at how to apply those concepts and multiple detection processes to the database itself. A major element of risk exposure in any organization is found inside the organization itself. The fact that people must be trusted when they are in administrative positions, and often have access to more information than they have an actual need to use. The very nature of database administration requires that the DBAs and managerial users have tremendous power over the data, and have the means and the opportunity to use that power in ways that might not be in accordance with the security policies in their organization.

The implementation of defense in depth in a database varies with the kind of data you store, as well as how that data is used in your day-to-day business process. It is important that the data needed to run your organization remains accessible in a realistic way for authorized users. Any security strategy that impedes data access by key decision makers at critical times may prove more damaging to your organization that the exposure of that data to others can be. A critical

mission of the data security team is to obtain a clear understanding of what your data is, who is allowed to use it, and how they are allowed to use it. In defining strategies to implement, which will increase confidence in the data integrity, many issues will be discussed, and many techniques to implement them will be covered. This discussion will focus mainly on an Oracle database, but the strategies to be implemented are applicable to any database.

Key elements of database defense in depth:

1. Data Security Policies (authorized usage)
2. Access Control (roles & privileges)
3. Password Control (authentication and non-repudiation)
4. Account Control (creation & removal)
5. Setting up Auditing
6. Monitoring Audit Logs
7. Periodic Analysis of Changes or "Red Flag" Situations
8. Setting Tripwires and Traps
9. Patching and Testing

1. Data Security Policies (authorized usage)

The security policy is at the heart of the design of data security. It should be short, clear, and specific. This document is used to define the purposes to which your data may be put, the terms of access, and the acceptable usage of your systems. It can clearly state that users are provided access to specific types of confidential information as part of the performance of their assigned duties, as well as specifying the types of actions that are prohibited. It should clearly warn the users that their actions will be monitored for security, and that violations of the acceptable usage policies can and will result in unfavorable actions (from reprimands to prosecution as warranted by the sensitivity of the information.)

2. Access Control (roles & privileges)

A key element in your database design is who can access what kind of data, and how. This is defined by your business practices and your data security policies. Usually there will be a job with a set of functions to be performed, each with it's own "need to know" in the data. These functions and access items can be controlled using the database roles, privileges and standard database account security practices. The details of how to create roles and assign privileges to them are discussed at length in standard DBA manuals (Ref. 1 - pages 296-334), (Ref. 9)

- REMEMBER THAT YOUR APPLICATIONS ARE NOT THE ONLY WAY TO CONNECT TO AND EXTRACT DATA FROM THE DATABASE! Security auditors are not the only ones who attempt to bypass your applications and access your data directly. If a user knows a valid user ID in the database, then any method of connecting to the database can be used to view the data. The default amount of information someone should be able to read in your database if they request it with minimal authentication is NONE.

CRITICAL ISSUE - One location that most folks don't talk about and Oracle tap dances around is a sweet little view called SYS.LINK$. If your database has a "link" (a live data connection to share information) setup to another database, that link is recorded here, along with the time it was created, the username used for the connection AND **THE PASSWORD TO THE ACCOUNT IN CLEAR TEXT**. The accounts used for database links are usually privileged accounts, so the passwords for those accounts are supposed to be the most guarded. Oracle has of course known about this **MASSIVE SECURITY HOLE** for years and has "enhancement requests" logged to eventually fix this. (for more information, see Ref. 12 - Oracle Metalink and do lookups on the status of Bugs or Enhancements numbered 1828368 and 70740). The data is stored unencrypted so it can be sent in unencrypted form to the remote database (which would not accept an encrypted password), as there currently is no encrypt/decrypt functionality, which is inherent in database links. This means that ANY USER WITH "SELECT ANY TABLE" PRIVLEGE CAN SEE THESE PASSWORDS!!!

- Remember that the database assumes the MOST access will be granted based on the roles and privileges allowed. If the user has multiple roles assigned to them, and those roles are active, then any privilege in any of those roles can be used.

- Consider having roles that are assigned, but are NON-DEFAULT roles, making them inactive at startup. You also want to have non-default roles PASSWORD PROTECTED so the user can't turn them on themselves. Your application software itself can set the role as active and enter the role password (set role ROLENAME identified by PASSWORD), so that the user has no knowledge of what the role and password are. The passwords for the roles can be stored in a restricted or encrypted area to prevent user access. Even the system privilege "CONNECT" that allows the initial ability to even say "hello" to the database can be turned on and off by the application software under your control. This hinders access to accounts using "data browser" tools.

- Try to not grant privileges directly to user accounts, but control them under roles. If a user's job requires that they have access to certain data at a higher level than others, create a new role for those functions and grant the role to the user. This allows isolation of the access required for a specific job function, so it can be granted, or revoked independently from other access privileges.

- Database accounts which have "sysdba" privileges should be few, and should always be monitored. These accounts are "super users" and can have access to any data and any file in the system.

- Be wary of any privilege that contains the word ANY as these allow users to create and manipulate objects in others schemas. It is like giving write permission on other people's directories in the operating system (select grantee, privilege from dba_sys_privs where privilege like '%ANY%'). Review these periodically to see if any new or unusual access grants have appeared. (Especially "SELECT ANY TABLE" as mentioned above.)

- Be stingy with grants to PUBLIC, since they are indeed public in nature, and are usable by anyone who can connect to the database. Be certain that any information that is accessible by PUBLIC is really something you don't mind the world seeing.

- BE SPECIFIC WHEN GRANTING PRIVILEGES! When a new object is created, be wary of developer's and DBA's shortcuts. A common practice is to create a new object (like a table), and then say "GRANT ALL ON table_name TO role_name or public". The GRANT ALL statement allows users not only to do the basic row level data manipulation functions of SELECT, INSERT, DELETE, and UPDATE, but also automatically allows data definition functions like ALTER (change the data storage structure and columns in the table), INDEX (add, modify and drop indices on the table), REFERENCES (foreign key controls), EXECUTE (the ability to run packages and procedures), and can also include READ (a directory access control.)

A more global access control can be implemented in Oracle using the "PRODUCT USER PROFILES". This allows you to specify specific commands and features that will be disabled for specified users and products. (Ref. 1 - Page 328 Limiting Available Commands: Product User Profiles) and (Ref. 3). This can be used to globally turn off features such as the HOST option (allowing operating system access.) The table that contains the restrictions is created by running the PBPBLD.SQL file as the SYSTEM user. Once created it can be populated with rows specifying the features you want to disable.

Example:

PRODUCT:	SQL*Plus
USERID:	%
ATTRIBUTE:	HOST
CHAR VALUE:	DISABLED

3. Password Control (authentication and non-repudiation)

To protect password confidentiality, Oracle allows you to encrypt passwords during network (client/server and server/server) connections. (Ref. 13) If you enable this functionality on the client and server machines, Oracle encrypts passwords using a modified DES (Data Encryption Standard) algorithm before sending them across the network. It is strongly recommended that you enable password encryption for connections to protect your passwords from network intrusion.

To encrypt net packets you need to install Oracle Advance Security option. Then it's a matter of configuring the following sqlnet.ora parameters on the client and the server

client
=======
sqlnet.crypto_seed
sqlnet.encryption_client
sqlnet.encryption_types_client

Server
======
sqlnet.crypto_seed
sqlnet.encryption_server
sqlnet.encryption_types_server
To what values you wish to use.

You should also seek out every default password used by Oracle and change it. This includes SYS, SYSTEM, and if running the Oracle Applications system, such database accounts as APPS (which controls all data access), and the accounts for each application you have loaded. The default password for the Oracle Applications APPS account is APPS. The other application installation passwords are similarly difficult to figure out.

Oracle8 and above also have more granular control of database password complexity, expiration and reuse. (See Ref. 1 - pages 308-318 Password Management) Enabling this feature will limit the ability of lazy users to use password shortcuts, which weaken your database security.

4. Account Control (creation & removal)

When creating user accounts, always use the "principal of least privilege". Authorization of the account creation should come from the responsible management level that controls the task that the user will perform. Document the request and the creation of the account. Document any changes that are authorized to the access granted to the account (roles set or changed.) Implement a plan to re-evaluate authorized access for a user when they switch positions and the needs of their job have changed. Implement a process for immediate lockout of the user when a termination has been implemented. Plan for what to do with the database objects and files owned by the user when their account has been terminated.

If possible, you may want to consider an automated system that is linked to the employee work status database. Any time someone changes their organizational role (promoted, transferred), or their status (left the organization, changed from temp to permanent, etc.), or their manager, an automated request should be sent to review the roles assigned to the user, and any applications or systems they have access to. **ACTION ITEM:** The manager of any employee should have a method of tracking what databases, systems, and applications each of their employees are authorized access to, what their duties related to those systems are, and what accounts and privileges they have been given in order to complete those tasks. Such a tracking system allows quicker lockout if a security situation arises, as well as easier setup of a new employee hired for the same functions.

5. Setting up Auditing

Auditing is a feature that allows you to track what people are actually doing in your database. (Introductory information is in Ref. 6). In large and active systems this is useful, but must be done with control and care as the data grows quickly.

Auditing is setup by creating the Audit Trail Views using the Oracle provided script CATAUDIT.SQL (Ref. 5 - Creating and Deleting the Database Audit Trail Views). If these views do not exist, then no auditing can be performed.

Auditing can be as specific or as broad as you desire. It can be used to watch for table deletion or truncation, alterations to "fixed" information, attempts to read from sensitive tables, etc. You can record user names, session ID, terminal, object accessed, operation attempted, completion codes, date and time stamps, as well as the system privileges used to perform the action (Ref. 6 - Audit Records and the Audit Trail). These can be recorded BY SESSION (one record per session, per object accessed per action), or BY ACCESS (one record for each access or auditable action attempted). (Ref. 5 - Auditing BY SESSION versus BY ACCESS). Each has its place, based on the amount of information you need to collect about the object, or the level of suspicion you have about what the specific user may be doing.

Do not make the mistake of always auditing everything, and at the most detailed level possible all of the time. All this will do is ensure that your users will be impacted hugely in their daily operations, and the audit data will rapidly exceed the storage capacity of your systems, growing to be many times the size of your valid data. Audit functions should be specific, and limited to monitoring the data and actions necessary to keep your system secure. The types of actions being audited, and the method used will change regularly based on any current changes in personnel, suspicious activities, or the need to look more closely at a specific item that was "red flagged".

One useful element that is often overlooked in audits is the field "terminal identifier" this will often contain the host name, or the IP address of the system from which the action was executed. If each user has a named terminal or PC, it can be enlightening to know when a userID has been used from a terminal that is not their own. That can often be an indication that someone is misusing the account of another person.

If there is a known time frame that your users are not normally accessing the system (say the hours of midnight to 6 AM), you might want to set a DBMS_JOB to audit connections and disconnections to the database during these "off hours". Such connections may be related to activity the user may not want to perform during normal business hours.

If users are supposed to work from their own systems, and the system names and IP addresses for each user are known, it should be simple to query the audit trail for anomalies where a user is connecting to a database from a terminal that is not their own. This could be an indication of login abuse.

BE SURE TO AUDIT ACCESS TO THE SYS.LINK$ VIEW.

6. Monitoring Audit Logs

By default, your SYS.AUD$ table will be created with up to 99 extents of only 10K each (Ref. 5 - Controlling the Growth and Size of the Audit Trail). This is not much to work with, and the storage parameters can be changed to increase the amount of storage allocated and growth rate

(any option except INITIAL storage parameter). As you experiment with your auditing, keep careful track of how quickly this grows under "normal" audit conditions. If it grows too quickly, you will need additional restrictions on the types of auditing being done.

Oracle has occasionally suggested the audit trails be purged. As with any log files, a decision needs to be made as to whether to export the current data to a file and place it to tape, extract the records to a new table, or even transmit the audit data to a completely separate database on a separate machine before taking any destructive actions. Which procedure you use is based on the sensitivity of the data in the system, the legal requirements you may be required to obey, and whether the audit is being done as potential evidence of criminal wrongdoing.

7. Periodic Analysis of Changes or "Red Flag" Situations

Use the Unix "GREP" command, or the advanced features of the Windows "FIND" command to scan your system for a known password. While these scans can take a large amount of time, they can surprise you with their results. You can occasionally find passwords hard coded into scripts, left in text files or mail messages, and even echoed into log files output from a script. Do such searches during off hours to limit impact on your users, and the visibility of the fact that you do such searches. You want to be subtle with these types of searches. (This is briefly mentioned in Ref. 11 in the section on "Overlooked Security".)

CRITICAL ISSUE - At one site that did the kind of text string search on their NT servers looking for files that might contain an administrator password, they located the password in the clear in the SQLNET.LOG files in locations where SQL files had been executed. That file logs errors in SQL connections. The passwords were in the clear because people had made simple typing errors when trying to connect to a database, and the error was written to the log file (e.g. CONNECT MYID/MYPASSWORD@MYDABATASE instead of CONNECT MYID/MYPASSWORD@MYDATABASE). Technically, Oracle didn't write the password in the clear here, it just wrote a bad text string. Either way, it compromised the security of a password.

Database Views to peruse for anomalies: DBA_ROLES, DBA_ROLE_PRIVS, DBA_TAB_PRIVS, DBA_COL_PRIVS, DBA_TAB_PRIVS, ROLE_ROLE_PRIVS, ROLE_SYS_PRIVS, ROLE_TAB_PRIVS

Queries that may provide indications of excessive rights and should be run periodically. Look for privileges out of scope for DBAs, and your known applications and roles. Look for "super user" privileges assigned to users, or roles that don't need them.

SELECT GRANTEE, PRIVILEGE FROM DBA_SYS_PRIVS WHERE PRIVILEGE LIKE '%ANY%'; (limited list of super user rights, that cross ownership boundaries. Few if any users should have any of these privileges.)

SELECT ROLE, PRIVILEGE, ADMIN_OPTION FROM ROLE_SYS_PRIVS; (look for anyone other than a DBA who has the ADMIN_OPTION set, since that allows them to grant the same privilege to others.)

SELECT GRANTEE, OWNER, TABLE_NAME, GRANTOR, PRIVILEGE, GRANTABLE FROM DBA_TAB_PRIVS; (should be a short list of directly granted privs to users.)

8. Setting Tripwires and Traps

Once you gain familiarity with the normal results from your "sniffer" queries, you have the option of automating some of the searches. Among the most useful are setting up "TRIGGERS" and "DBMS_JOBS" to do some of your looking for you.

If no user except FRED has the DBA role, set a trigger on the DBA_ROLES to insert a record into a tracking table when anyone except FRED is assigned a DBA role.

Setup jobs that run daily or weekly and execute the searches you run most often. Have them mail you the output, or place it into a restricted file or table for later review.

If you are about to terminate an employee, and they don't know it yet, consider setting an audit function on that user ID for a period of time before they actually leave. This may give an indication of any last minute "housekeeping" they might try to do, or attempts to sack and pillage your data.

You will probably want to have very few users able to use the "DELETE ANY TABLE" privilege, since it will allow deletion of records from the SYS.AUD$.

Audit changes to the audit trail itself using "AUDIT INSERT, UPDATE, DELETE ON SYS.AUD$ BY ACCESS;"

Consider adding a trigger to monitor the SYS.AUD$ table (Ref. 5 - Auditing Through Database Triggers). This way if someone does have access to edit or delete items from that table, the act of changing it can be recorded in a secondary area. By having different monitoring facilities, watching in different ways, and storing their results in different areas, you vastly increase the probability that an intruder trying to cover their tracks may miss a few of their footprints.

Some commercial products are emerging that make claims to automate the audit trail analysis for you. Beware. They are indeed useful, but they are only as effective as what you have told them to look for. Like the analysis engines used for firewall analysis, these tools can and will overlook items they have not been told to look for, sometimes causing a false sense of security in an administrator. If you use such tools, remember that sometimes it takes a human eye and reasoning to see things that an automated system can miss.

Automated audit analysis systems can also take quite elegant forms, exceeding the requirements of most organizations. One such "paranoid" system is discussed in the paper "DAIS: A Real-time Data Attack Isolation System for Commercial Database Applications" (Ref. 7) This system

calculates a rating of how unusual the individual and cumulative actions of a user in the database is, and sets up a pseudo environment similar to the "fishbowl" concept in order to allow isolation and examination of the suspect user's actions. This concept is probably overkill for many organizations, but may have interesting possibilities in commercial security software.

9. Patching and Testing

Patches to code are like patches on pants. They may cover the hole you see, but they may cause additional problems you didn't expect. Remember that all coding is done by people, and even with excellent coding and testing practices, errors can occur. The more complicated the system you are trying to maintain, the more likely there will be unexpected consequences from a patch. TEST TEST TEST!

Keep track of security issues and patches applicable to your environment by multiple means. Oracle announces SECURITY ALERTS and patch availability on the Oracle Technology Network website (Ref. 4), on their METALINK technical site (Ref. 12 - www.metalink.oracle.com), and via third party security monitoring news services like SANS Security Alert Consensus newsletter (Ref. 14 http://www.networkcomputing.com/consensus/). By the time you read about it, someone has already exploited it somewhere!

Not all patches are applicable to your environment. Verify the pre-requisites and understand the scope of a patch before you apply it to your system. If you don't need it now, you probably will still get it later when the next major release comes out, and by that time, the integration testing of the rolled up patches will be more thorough.

Not all patches are error free! You need several database instances to test in. Have an instance (COMP or DEV) for your developers (patches up to date with PROD) to write code in and check that it works in theory. Have an instance for testing new patches that will not interfere with your developers (COPY OF PROD) and use this copy to test the installation procedures for the patches to make sure it works smoothly. Have a place to run an integration test (INTT) to make sure basic process are still running after adding newly developed code, or installing patches.

If you need to let your boss know the "security level" of your version of Oracle, you need to know some of the standards applicable. Many of these are listed in the Oracle Security Handbook (Ref. 2), on page 539. The fact that a product CAN meet these standards, does not imply that the way an organization has implemented that release of the product is in any way secure. Constant vigilance is needed. (Examples are Oracle 7, Release 7.3.4 can meet ITSEC E3/F-C2, and Oracle8, Release 8.1.6 can meet Common Criteria level EAL-4).

The "C2" standard is from the "Trusted Computer System Evaluation Criteria (TCSEC), also known commonly as the "Orange Book" standards, from 1985 (yes, 16 years ago.) While a valid standard in it's day, it has served its purpose, and was retired at the end of year 2000 according to the training I received at the SANS conference in December of that year. The TCSEC (C-2) standards were designed and were valid for stand-alone computers. Once connected to a network, the TCSEC evaluation criteria were invalid. The U.S. Trusted Product Evaluation Program (TPEP) and the Trust Technology Assessment Program (TPAP) will no longer accept

new evaluations based upon TCSEC starting 1 February 1999. The "Common Criteria" (CC) standards must used, and by the end of year 2001, all formerly evaluated products against TCSEC will become obsolete or must be reevaluated by the CC process. (Ref. 16)

In 1987, the Trusted Network Interpretation (TNI) was added to interpret the TCSEC for trusted computing in a networked environment.

These standards were OK, but they were mostly about confidentiality issues, and not about integrity and availability. They were also U.S. Government standards, and did not get much international support. They were added to the Federal Information Processing Standards (FIPS) program run by NIST, but those standards have not been updated to reflect the new technical realities. They were replaced by the Common Criteria Project.

The criteria effort currently focused on is the International Technology Security Evaluation Criteria (ITSEC). This combined TCSEC with international input, integrity, and availability. The "Common Criteria" (CC) project can be used to develop the concept of "Industry Best Practices" that are commonly followed in implementing varied multi-layered security functionality in a target environment. The Common Criteria is replacing the national criteria with a worldwide set accepted by the International Standards Organization (ISO). CC provides seven assurance levels labeled EAL1 to EAL7. This project is sponsored by the U.S., Canada, France, Germany, the United Kingdom and the Netherlands. The latest version is ISO 15408 - CC v2.0. The EAL -3 is roughly the functional equivalent of the old C-2 standard, but I don't think that is what we are intending to stop at.

A highly regarded standards document that is gaining critical acclaim for comprehensiveness and quality is from the United Kingdom. British Standard 7799 (BS7799) was developed in 1995, and the latest update is 1999. It is designed to provide a comprehensive coverage of controls and industry best practices for infosec for industry and commerce. That document is one most likely to for many organizations to work towards.

More information about these and other security standards can be obtained at:

http://csrc.nist.gov/cc
http://www.bsi.org.uk/bsi/products/msr/bs7799/index.xhtml
http://www.sans.org/infosecFAQ/securitybasics/criteria.htm
http://www.sans.org/infosecFAQ/policy/standards.htm
http://www.sans.org/infosecFAQ/standards/ISO17799.htm
http://www.sans.org/infosecFAQ/policy/standardization.htm
http://www.sans.org/infosecFAQ/policy/matures.htm
http://www.sans.org/infosecFAQ/start/sec_assistance.htm
http://www.nstissc.gov

SUMMARY:

While there are many tools available to the information security specialist in limiting access to the database, tracking questionable activities, lowering the visibility of sensitive information, and

hardening the system, what you do with them is up to you. Out of the box, the database, like most operating systems and firewalls, is NOT secure, and will not be configured to your needs. Planning and research is needed to design your system tightly from the beginning, and considerable effort may be expended in forcing such security onto an already bad design. Use every upgrade and redesign opportunity to install new phases of security quietly. Constantly monitor changes and have more than one method of looking. Remember that not all technicians and DBAs are perfect, and mistakes can happen (sometimes on purpose too!)

Defense in Depth At Every Layer!

References:

1.
Oracle8 DBA Handbook
Oracle Press
Osbourne / McGraw-Hill
ISBN 0-07-882406-0
Chapter 9 - "Database Security and Auditing"

2.
Oracle Security Handbook
Osbourne Press
Chapter 17 - "Hacker Proofing Your Database"
http://www.osborne.com/database_erp/0072133252/0072133252_ch17.pdf

3.
SQL*Plus® User's Guide and Reference
Security (PRODUCT_USER_PROFILE Table / Roles)
http://storacle.princeton.edu:9001/oracle8-doc/server.805/a53717/ape.htm

4.
Oracle Technology Network
Security Alerts
http://otn.oracle.com/deploy/security/alerts.htm

5.
Oracle8i Administrator's Guide
Release 8.1.5 - A67772-01
25. Auditing Database Use
http://technet.oracle.com/doc/server.815/a67772/audit.htm

6.
Oracle8i Concepts
Release 8.1.5 - A67781-01
31. Auditing
http://technet.oracle.com/doc/server.815/a67781/c27audit.htm

7.
DAIS: A Real-time Data Attack Isolation System for Commercial Database Applications

Peng Liu, Department of Information Systems, UMBC
http://www.acsac.org/2001/papers/44.pdf

Research Papers related to Databases on http://www.sans.org/giactc/GSEC.htm:

8.
Database Security in High Risk Environments
Joaquin A. Trinanes
September 14, 2001
http://www.sans.org/infosecFAQ/appsec/db_sec.htm

9.
An Overview of Oracle Database Security Features
Lorraina Hazel, CNE
May 13, 2001
http://www.sans.org/infosecFAQ/appsec/oracle.htm

10.
Securing Databases
Paul Carmichael
April 9, 2001
http://www.sans.org/infosecFAQ/appsec/database.htm

11.
Web Application and Databases Security
Darrell E. Landrum
April 2, 2001
http://www.sans.org/infosecFAQ/securitybasics/web_app.htm

12.
HTTP://WWW.METALINK.ORACLE.COM
Bugs & Patches

13.
http://metalink.oracle.com/metalink/plsql/ml2_documents.showFOR?p_id=309886.999&p_show
Header=1&p_showHelp=1

14.
SANS Security Alert Consensus Newsletter http://www.networkcomputing.com/consensus/

15.
Hack Proofing Oracle
Howard Smith, Oracle Corporation UK Limited
http://otn.oracle.com/deploy/security/pdf/oow00/orahack.pdf

16.
(K. Minihan. Advisory Memorandum on the Transition from the Trusted Computer System Evaluation Criteria to the International Common Criteria for Information Technology Security Evaluation. Mar 99, NSTISSAM COMPUSEC/1-99.)