Considerations for Securing Data in Oracle Databases

GIAC Security Essentials Certification
Practical Assignment Version 1.4b Option 1

Abdur-Rahman Husain, OCP

April 23, 2004

Abstract

Data represents one of the most valuable assets of an organization; some may say the most valuable. Most organizations store their valuable data in databases. In the security industry in general, a lot of attention is given to securing networks and hosts, less on securing applications and databases. However, in keeping with the principal of defense-in-depth, database security is a vital component of the overall security strategy of an organization.

This paper attempts to highlight the security considerations in securing data within Oracle databases. Oracle provides a variety of features to ensure the availability, integrity and accessibility of the data. This paper will present a number of elements related to Oracle databases, which individually mitigate a subset of security threats, but when applied together, provide a more comprehensive level of protection. These elements or layers of security include software installation, authentication, access control, encryption, source code control, auditing, penetration testing, availability, application interface, human interface and security plan. Although there are many considerations in securing the network related to the database and the database host, these issues are generally better understood and there is a fair amount of literature that covers these topics so they will not be discussed in this paper.

Most of the security elements are available across recent Oracle versions. However, Oracle 9i Release 2 Enterprise Edition is the reference database version for this paper.

Software Installation

In keeping with the general trend in the software industry to provide better "out-of-the-box" default security, Oracle has implemented better default security for a database installed with the Oracle Database Configuration Assistant (DBCA). There are four main considerations in the installation of an Oracle database.

1. Minimizing the database components that are installed to only those that are required [1]. Oracle offers many features that are available at installation time. It is recommended to install only those that are required immediately. Installing unused features only increases the administrative burden and increases the security risk. The DBCA makes it very simple to install any additional feature on an existing installation if required as well as to remove any existing feature which is no longer required [1].

2. Controlling usernames and passwords, especially those with administrative privileges [1]. By default, installing a database with the DBCA locks and expires all default accounts that are created by the installer except for the SYS, SYSTEM, DBSNMP and SCOTT accounts [1]. At installation time, the administrator is prompted to provide a

password for SYS and SYSTEM, the two main administrative accounts. However, DBSNMP and SCOTT maintain the default password and are left open by default [1]. This is a very common and prevalent security hole. These default passwords should be changed at installation time using a strong password [11]. Any time a default account is unlocked and opened, the default password should be changed accordingly. It is important to note that some of these restrictions are not enforced when the database is installed using script files rather than the DBCA.

3. Using a standard industry benchmark. There are several industry benchmarks and checklists for Oracle installations available freely on the internet. Using such standards is an excellent starting point for securing an Oracle installation. Examples of these benchmarks and checklists are available at http://www.sans.org/score/checklists/Oracle_Database_Checklist.pdf as well as http://www.cisecurity.org/bench_oracle.html [1]. It is important to note that these checklists are a starting point only and should be adapted to individual installations.

4. Keeping up to date on security vulnerabilities and patches [1]. This is one of the keys to maintaining a secure installation and protecting against buffer overflow vulnerabilities and other software bugs. Oracle provides a full listing of security alerts at http://otn.oracle.com/deploy/security/alerts.htm as well as a way to register for e-mail security alerts. Other security sites such as http://www.securityfocus.com/bid/vendor/ should also be monitored regularly for security advisories. Applying security patches should be done in as timely and safe a manner as possible. As usual, it is good practice to implement the patches in a test environment before applying them to a production environment in order to verify that the patch does not have any unforeseen side effects.

Authentication

Any authentication system seeks to verify the identity of the user; to make sure that the user is who he claims to be [2]. Oracle provides a basic form of authentication to the database via usernames and passwords. The password management functionality allows the administrator to set password policies such as password format, password expiration, and password re-use. The password policy is implemented through profiles which are a named set of resource restrictions that can be assigned to user accounts. One of the elements of a profile is the password policy. By default, every user created in the database is assigned the DEFAULT profile. A good strategy is to create different profiles based on the requirements for different types of user accounts and assign these

[1] This site requires a free registration and agreement on Terms of Use before the document can be downloaded.

explicitly to each user account. The user accounts themselves can be defined at the operating system level or in the database [2].

In order to handle the case of multi-tier web applications where users log in the to web or application tier which in turn logs in to the database with a single database user account, Oracle provides what is called proxy authentication [2]. This means that a users session information from the web or application tier – which identifies the user uniquely – can be attached to the database user account in a context which is then applied to any existing access control policy defined at the database level. Essentially, this allows some level of access control even for multi-tier web applications where the actual user's login is not known to the database.

For larger organizations that have investments in 3rd party authentication systems that are used centrally to authenticate users to all networked resources, Oracle provides the ability to leverage these systems through an additional product offering called Oracle Advanced Security [2]. With Oracle Advanced Security, authentication to the database can be done via mechanisms such as RADIUS, Kerberos, Smart Cards, Biometrics, Public Key Infrastructure (PKI) and Token Cards. This type of authentication makes a lot of sense in reducing the administrative overhead, having a consistent authentication policy across an environment and possibly even making life simpler for the end user with the single sign-on capabilities that some of these authentication mechanisms provide [2].

Access Control

Once a user has been authenticated to a database and a user session has been established, the problem now becomes how to limit that user to see only those database objects and that data that he is authorized to see [2]. This is the role of access control. Access control is the heart and soul of database security. It is comprised of two parts – managing privileges for users of the database and controlling access to database objects based on those privileges even down to the level of rows and columns of a particular table [2].

There are two types of privileges in an Oracle database – system and object privileges. System privileges refer to privileges at the entire database level such as backing up the database whereas object privileges refer to actions on particular database object such as being able to select from a particular table or execute a particular stored procedure [2]. From a security perspective, the rule of least privilege should be followed. This means that especially in a production environment, all privileges should be initially revoked from all users and roles and then assigned explicitly to users as they require them. Also, system privileges should be controlled very carefully in a production environment as opposed to a development environment where developers may be given these privileges more freely to, for example, create and drop objects as required.

Managing privileges for many users individually can become a large administrative burden. Fortunately Oracle provides the concept of roles which are basically a named group of privileges that can be assigned directly to users or even to other roles. These roles can be enabled and disabled on demand and can be protected by passwords or even procedural logic. An administrator can dynamically add or remove privileges from a role and that change will take effect on all user accounts that have been assigned that role. This gives the administrator much more manageability on user privileges and therefore better access control administration. In cases where there are multiple Oracle databases, Oracle provides global and enterprise roles which can be applied across different databases [2].

Database views and stored procedures can also be used to enhance access control. The idea is to grant privileges on views and stored procedures without giving any access to the underlying database objects [2]. Thus, if a user has the execute privilege on a stored procedure that updates specific columns of a table, the user cannot update other columns of that table because that user account does not have any privileges on the table itself. Similarly, a select privilege on a view will give access to those columns and rows in the view without giving access to the underlying tables. Oracle extends this concept with Virtual Private Database (VPD). VPD simplifies the task of creating multiple access control policies on the same database object so that the appropriate policy is applied based on the security context of a particular user at the row level [10]. This functionality is especially useful when there are diverse applications or user communities accessing the same base database objects. With VPD an administrator is required to build the access policy, so there is a degree of programming involved in order to implement this properly.

Oracle does supply a pre-built VPD solution as a separate product called Oracle Label Security which requires only customization of existing policies rather than building them from scratch [2].

Encryption

Encryption can be applied to data at the network level in the transport of data between a client application and the database. This topic would be covered under database networking which is not the scope of this paper [6].

Encryption, as applied to stored data, supplements access control and provides a further layer of protection [6]. It helps to ensure that data is accessible to only the holder of an electronic key that is used to encrypt and decrypt the data. Encryption of data inside the database can also be used to “hide” data from privileged users such as DBAs. Encryption provides a measure of data integrity. It does not guarantee that data cannot be changed, but it does provide a method to detect if unauthorized changes have been made [6].

Oracle provides the standard DBMS_OBFUSCATION_TOOLKIT package to implement encryption of data within the database. This package supports Data Encryption Standard (DES) and Triple DES (3DES) as well as MD5 checksumming which provides the ability to digitally sign data so that any unauthorized change can be detected [7].

There are several issues in implementing encryption within the database.

1. Encryption is a resource intensive operation. Care should be taken to encrypt only very sensitive data such as credit card information which is highly sensitive and not accessed frequently [6]. Data access controls are much more efficient and should be implemented first before determining if a further layer of security is really required.

2. Key management is not provided by Oracle. The encryption key can be generated by using the GetKey function of the DBMS_OBFUSCATION_TOOLKIT package which complies with the Federal Information Processing Standards (FIPS)-140. However the transmission of the key over the network and its storage are not provided for by Oracle [7]. These can be implemented on a custom basis or they can be delivered by 3rd party products that are compatible with Oracle.

3. Encryption keys, like passwords, should be changed regularly. This can have an impact on database availability if large amounts of data need to be decrypted and encrypted again using new keys [6].

One of the very appropriate uses of encryption is to protect the storage of database backups [6]. Backup files are usually not accessed frequently so the encryption overhead is not significant. Backups can be stored on the database host or on other media such as magnetic tape and are very frequently stored offsite. Backup files can also be used to transport databases from one location to another – for example from a production system to a test system. In all cases, encryption of the backup files provides an extra layer of protection for files that otherwise may not be protected by stringent access controls even though the data contained in the files is just as valuable as that in the online database. Oracle does not provide an encryption technology for such files outside the database, but there are 3rd party products that fill this need.

Availability

Availability is a very important aspect of security and addresses a few different security risks. Availability can mean protection against denial-of-service type attacks where database resources are tied up by rogue process and therefore the database becomes unavailable to legitimate users [2]. It can also mean the ability to recover from an attack or debilitating compromise within an acceptable timeframe.

In order to protect database resources from rogue processes, Oracle provides both storage and CPU constraints that can be applied to a user account [2]. Storage constraints limit users to defined quotas on specific tablespaces thus creating a maximum limit to the amount of data any one user can store in the database. Resource constraints are implemented using profiles. We have already seen how profiles are used in setting password policies. Profiles also allow for setting limits on a user account's use of CPU, logical I/O (i.e. how much data can be processed), idle time and connect time for a user session [3]. Different profiles can be created for different types of users within the database based on their need.

The ability to recover from a data corruption, an unauthorized modification of the database or any security violation that renders the database unusable is provided by a robust backup and recovery strategy. Oracle provides several tools and techniques to implement this. Several types of backups are available.

1. Cold Backup. This means that the all database files are backed up to another location while the database is shut down. The advantage of this type of backup is that it is fairly easy to manage. However there are several disadvantages to the cold backup because it requires the database to be shut down, the entire database must be backed up not a portion of it and recovery is only until the time that the backup was taken.

2. Hot Backup. This is a backup of database files while the database is open and available for users. This type of backup requires the database to be in archive log mode so that all database transactions are preserved in archived redo log files. The advantages of this type of backup is that the database is available at all times, there is flexibility to backup up portions of the database at any one time and there are many more recovery options. The disadvantage for hot backups is that they require much more management and expertise for recovery.

3. Export. This is a logical data backup that backs up both the structure and data within a database. The advantages of export include flexibility to backup portions of the database and it is relatively simple. The disadvantages are that it records the database at a particular point in time and therefore does not have many recovery options.

A good database backup strategy requires elements of all three types of backup. A true production database should always be running in archive log mode and should always include hot backups.

A supplemental tool for backup and recovery provided by Oracle is Data Guard which allows for one or more standby databases to be configured for a particular primary database. The basic mechanism used to update these standby databases is to apply the transaction logs from the primary to the standby thereby keeping them in sync. It is possible to keep both the primary and standby in sync in real time, however most installations are usually slightly out of sync. Data Guard enables a very quick way to recover from the unavailability of the primary database. It can be configured so that primary and secondary databases can switch roles with minimum or no loss of data.

Source Code Control

By default all code from packages, stored procedures, functions and triggers is stored in the Oracle data dictionary as clear text. Normally, access control should limit access to the data dictionary. However, just as encryption of data provides an extra layer of security to sensitive data, so can encryption of source code in the database provide an extra layer of security to the sensitive programming logic in the database.

Oracle provides a wrap utility that encrypts database source code in byte code. The wrap utility accepts a simple text file containing the source code as input and then produces a file with byte code as output that can then be loaded in the database [4]. There are a couple of issues associated with “wrapping” source code.

1. The resulting byte code is two or two and a half times the size of the original code so it has an impact on storage and memory usage. However, there is no effect on the execution time [4].

2. The wrap utility is very sensitive to the version of the Oracle database. If the same piece of code is being deployed to multiple versions of Oracle, it should be wrapped by the wrap utility of each version respectively [4].

Auditing

No security strategy is complete without having some way to detect and log security violations or violation attempts. Oracle’s default auditing feature is very comprehensive and full-featured [5]. It is possible to audit statements in the database (e.g. select, delete), system privileges (e.g. create tablespace), any activity on any database object or any activity by any user of the database. Another element of the audit capability is the ability to audit both successful and unsuccessful activities [2]. There are two possible locations for the storage of audit trails – the AUD$ table in the SYSTEM tablespace and in the operating system for selected operating systems only. There are several issues to consider in auditing.

1. Oracle’s auditing feature is quite efficient in that it executes only once together with the audited statement [5]. However, depending on the type of auditing that is configured as well as the general volume of audited transactions, the audit trail can fill up fast. If the audit trail is implemented in the database, the risk is that the SYSTEM tablespace will fill up and render the database unusable. It goes without saying that the audit trail needs to be monitored very carefully and possibly moved to another area for reporting purposes regularly [5].

2. The audit options are vast. It is recommended to audit selectively so that the audit data itself is meaningful in detecting any data access abuses or security violations. Too much audit data is hard to analyze and defeats the purpose of auditing in the first place [5].

3. Auditing is the best way to hold database administrators and privileged users accountable for their actions on the database [6].

4. Ignorance of the auditing capabilities in Oracle very often leads application developers to develop their own audit functionality. Using the functionality that is there will provide a quicker development time as well as providing a more robust and tested auditing facility.

Oracle’s regular auditing feature does not provide auditing at the row level intrinsically. However, there is a solution if row level auditing is required – i.e. if there is a requirement to record the actual data content that a particular user might have modified or even read. For DML operations – insert, update and delete – row level auditing can be implemented by system triggers. Each DML operation fires a before and after event which can be associated with a trigger. The logic of the trigger has to be developed programmatically – nothing is provided by Oracle [5]. SELECT statement row level auditing is provided by a database feature called Fine-Grained Auditing (FGA) which allows an administrator to define an audit policy when a particular column or row is accessed and read by any user [2].

Penetration Testing

Once the database has been installed and configured for security, it is a good idea to test the installation for vulnerabilities by performing penetration tests. This will help to validate the existing database security as well as point out any security vulnerabilities. It should be part of any security strategy to perform these penetration tests on a pre-determined schedule that is appropriate for the organization depending on the rate and amount of change in the database environment as well as the security environment.

Several “Oracle-aware” tools and scripts are available for penetration testing. These tools and scripts have several advantages over building your own. First of all they have been prepared by experts in the field and you can leverage this expertise by using them. They are kept up-to-date to detect the latest vulnerabilities as they are discovered. Secondly and maybe more importantly, they provide a different perspective on security from that within the organization. They are not “locked in” to a particular way of thinking about security and therefore have more of a chance to discover vulnerabilities than those who have actually implemented the security on a particular database. This is similar to a software developer who may not find bugs in software that he wrote as easily as an independent tester.

A whole list of penetration testing and security scanning tools for databases are available from http://www.securitywizardry.com/database.htm . Many of these tools are available freely for download.

Application Interface

The database is not a standalone entity. Data is written to and read from the database through an application interface. Therefore the discussion on securing a database is not complete without a discussion of some of the application elements that interact with the database. There are several application design decisions that impact the security of the database.

1. Authentication. Many applications implement custom authentication modules by creating tables and code that implement login functionality without using the built-in authentication methods provided by Oracle. This is usually a large security risk since it is more than likely that the custom authentication functionality is not as robust or secure as Oracle’s which has been tested and used by a large number of customers. Usually the password in such applications is stored as clear text since encryption is not a trivial process. Also, much of the access control functionality in Oracle is rendered useless because Oracle applies privileges and implements access controls that are based on users that it “knows” about; not on users who are defined as records in an application table. When multiple applications are using the same database, it is easier to provide single sign-on functionality and a common authentication process across applications when authentication is implemented in a standard way.

2. Access controls. In today’s reality, databases are accessed by many different applications and types of application clients. These can include web servers (ASP, JSP, CGI), application servers (.NET and J2EE), client applications (Visual Basic, Oracle Forms), reporting tools (Crystal Reports), administration tools (Oracle Enterprise Manager), OLAP tools, database to database links and more. In this case it makes sense to implement the access control logic as close to the data as possible so that the data security rules are implemented uniformly across all applications. Consider an application where all access control logic is defined in a J2EE

application and then a user requests a reporting tool to on the same database. Those J2EE application access control rules do not apply to the reporting tool.

3. Database packages and stored procedures. From a security perspective as well as a performance perspective it is recommended to hide the implementation of the database using database packages and stored procedures. Any interaction with the database – insert, update, delete and select – can and should be implemented at the database layer. Therefore an application does not need to know about which tables are in a database, it needs only to know the appropriate database package or procedure with the required parameters related to the action that it wants to perform. From an Oracle perspective, execution privileges can be given to procedures and packages, not on the underlying tables in the database. A security breach of the application will not reveal details about the database. An added benefit is that validation of all inputs can be done centrally at the database regardless of the application that is accessing the database even though validation should also take place at the application layer to minimize the security risk of SQL injection attacks [9].

Human Interface

The human interface is an essential ingredient of the security of any database whether it is Oracle or any other type of database. By necessity, Database Administrators (DBAs) are granted administration privileges that are required to administrator a database. Among other things, these administrative privileges provide the ability to start and shutdown a database, create, modify, drop database objects, backup and recover a database, create or modify users and passwords.

There are several approaches to mitigate the security risk inherent in such powerful privileges.

1. Screen prospective DBAs before they are hired or given the DBA job. This is by far the most important step to ensure that a trustworthy individual is given the “keys” to an organization’s valuable resource. Screening can include background checks, contacting references and previous employers.

2. Divide the roles and responsibilities so that all the power is not in hands of one individual. This division can be along the lines of applications, or DBA functions. This means that if a database hosts more than one application, a particular DBA will be responsible to administer the database schema related to that application. Similarly, DBA functions such as startup and shutdown, backup and recovery, user management can be divided and distributed to more than one individual.

3. As discussed before, encryption can provide protection for sensitive data so that it can be "hidden" even from someone with DBA privileges in the database.

4. Auditing provides the means to hold even DBAs accountable for their actions on a database by enabling the recording of DBA actions to an audit trail in the database or operating system [5]. As in other contexts, auditing is the best candidate function to be handled by an individual other than the one who performs regular DBA duties.

Security Plan

One of the more neglected elements in a security strategy for Oracle databases is a security plan. This consists of a document or set of documents that formalize all the decisions that a particular organization has taken with respect to the security elements involved in securing an Oracle database [8]. It can and should include those elements discussed in this paper as well as other relevant elements related to Oracle database security.

The security plan plays an important role in communicating to all concerned personnel, an organizational standard with respect to Oracle database security [8]. It provides a standard against which all Oracle database installations and related processes should comply. This relieves individuals from implementing security as they would see fit but which may not necessarily fit with a particular organization's security posture. One of the most important benefits of preparing a security plan is that it forces all involved personnel in an organization to think through the security process before implementing it, thereby greatly reducing the potential security risks that are inherent in an ad-hoc security strategy.

A basic outline of a security plan for Oracle is available at http://www.oreilly.com/catalog/orasec/chapter/ch07.html. This can be used as a starting point in creating a plan.

Conclusion

The challenge in securing a database is that data is valuable only if it is accurate and available to the right people at the right time while at the same time protecting it from unauthorized access and tampering.

Oracle is a very complex database management software. Each of the elements discussed in this paper require separate research to explore them in detail. However, if we understand the elements from a big picture perspective, we can then implement each one as a piece of the database security puzzle. It is important to be aware of and address each of these elements. As with any security strategy, there is no such thing as a bullet-proof database. However, when all these elements are planned and implemented according to the requirements of an organization the overall security risk is greatly reduced.

References

1. "Secure Configuration Guide for Oracle9iR2." June 2002. URL: http://otn.oracle.com/deploy/security/oracle9i/pdf/9ir2_checklist.pdf (9 April 2004).

2. "Oracle9i Security Overview Release 2 (9.2)." Oracle 9iR2 Manuals. URL: http://download.oracle.com/docs/cds/B10501_01.zip [2](12 January 2004).

3. "Concepts." Oracle 9iR2 Manuals. URL: http://download.oracle.com/docs/cds/B10501_01.zip [3](12 January 2004).

4. Burleson, Don. "Best Practices for using Wrap." 28 October 2003. URL: http://www.praetoriate.com/oracle_tips_dm_bp_wrap.htm (15 March 2004).

5. Finnigan, Pete. "Introduction to Simple Oracle Auditing." 29 April 2003. URL: http://www.securityfocus.com/infocus/1689 (9 March 2004).

6. "Encryption of Data at Rest."URL: http://www.appsecinc.com/presentations/Encryption_of_Data_at_Rest.pdf (9 March 2004)

7. Davidson, Mary Ann, Browder, Kristy, Helmann, John, Needham, Paul. "Database Encryption in Oracle9iR2." April 2003. URL: http://otn.oracle.com/deploy/security/oracle9ir2/pdf/dbcrypt9ir2.pdf (9 April 2004).

8. Theriault, Marlene, Heney, William. "Chapter 7. Developing a Database Security Plan." Oracle Security. October 1998. URL: http://www.oreilly.com/catalog/orasec/chapter/ch07.html (11 April 2004)

9. "Protecting Oracle Databases." URL: http://www.appsecinc.com/presentations/Protecting_Oracle_Databases_White_Paper.pdf (9 March 2004)

[2] Requires a free membership at Oracle Technology Network http://otn.oracle.com

[3] Requires a free membership at Oracle Technology Network http://otn.oracle.com

10. Browder, Kristy, Davidson, Mary Ann, Helmann, John, Needham, Paul. “The Virtual Private Database in Oracle9iR2.” January 2002. URL: http://otn.oracle.com/deploy/security/oracle9ir2/pdf/VPD9ir2twp.pdf (10 April 2004).

11. Westervelt, Robert. “Expert offers tips on securing Oracle databases.” 15 July 2003. URL: http://searchoracle.techtarget.com/originalContent/0,289142,sid41_gci914696,00.html (11 April 2004).