DATABASE SECURITY 101: PREVENTING ESPIONAGE AND SABOTAGE

Richard D. Newallis, SPRINT

ABSTRACT

Over the last few years, the amount of data stored in corporate databases has grown exponentially. The accessibility of this data has greatly increased the productivity of these organizations as well as the security risks that they face. The security of the data stored in our databases is one of the most important, yet, least talked about, issues that organizations face in implementing a database system. This paper will examine how important database security is and how it impacts every organization. Some of the most common security breaches in databases are identified as well as how they can be resolved. Additionally, simple "common sense" ways to ensure the database is properly accessed and the security features available in Oracle8 are examined. The paper provides the first step towards implementing security "best practices" and a database security methodology for an organization.

WHAT IS SECURITY?

Security is one of the most important and most assumed aspects of database technology and database administration. Everyone agrees it is important and most people think that theirs is adequate. One can equate security to insurance – I have never met anyone who is so rich that they could afford to go without it, and you never know how much you need until you need it. There is a simple equation for security: Critical Data + Vulnerability = Disaster. When most people think about security, they immediately think of a hacker breaking into their system over a modem and deleting their files. It would be nice if things were only this simple in reality. While this scenario is painful, it is nothing that a solid backup plan and a short downtime cannot fix. When looking at database security, we must look at making the data secure in every way from anything that effects its availability, privacy, or quality. The absolute worst nightmare is stolen or altered data. Deleted data is usually very obvious, but altered data can exist and damage the company for an undetermined length of time before it is discovered. Suddenly, the task and importance of securing the database is taking on a new level of importance and urgency. If you are not scared, nervous, or feeling slightly uneasy about this subject – you should be.

Our society is more dependent on information and data than ever before; which makes the issue of security all the more important to the gatekeeper or the skilled DBA. The purpose of this paper is to cause you to think about your database and about security, and hopefully to come away with one of two feelings, either: "I understand security, I test my security, and am very confident my databases are secure." Or "I now better understand security and I realize what I must do in order to make my databases more secure." First, we must examine the concept of security, whom we are protecting our databases from, and then identify the different types of threats and components of security. Then, it is possible to examine our databases with respect to each type of security, each type of threat, and our organization's needs for security in order to create a comprehensive security plan.

WHO IS A THREAT TO MY DATABASE?

The answer is simple: Everyone is a threat. Of course there are the stereotypical threats, such as hackers and disgruntled ex-employees. Depending on the type of system and database, these people can actually be the easiest to guard against, but they are only the tip of the iceberg. In order to secure a database, we must protect it from much less obvious threats such as "internal" and "white collar" threats. Who are these "less obvious" threats? It may be Chris in sales and marketing, selling off spatial data to competitors or other companies with similar market demographics to make a little extra money on the side. What about Betty in accounting? Is she selling off financial info, customer or credit data? Lists of good or bad accounts? Who has access to product plans or campaign strategies? Unscrupulous companies will pay handsomely for converted product knowledge or information. Who in your organization could cause something that would adversely affect the stock price? Could they be selling short in the open market? The possible combination of motives and opportunities is virtually limitless. Now we can factor in all the possibilities where the motive is pure and honorable. These are where people have good intentions and motives, yet do not have a full understanding of what they are doing, and their actions negatively impact the company in some manner. This may be as simple as sharing a password so a co-worker can get something done. It may be running an update query without a proper where clause or no where clause at all. Regardless of their intent, the effects of their actions have a negative consequence for the organization. Who are these people? They can be a friend, a lunch partner, or the shortstop on the company softball team. When it comes to security, literally everyone in the organization is a threat.

TYPES OF THREATS

There are several attributes to consider in classifying a threat. It is vital when creating a secure environment that the DBA consider each combination of possible attributes and looks for how his or her system might be vulnerable to such a threat. These attributes are point of origin, motive, and form.

Attribute	Possibilities
Point of Origin	External or Internal - to the organization
Motive	Intentional or Careless
Form	Destructive (obvious) / Sabotage (hidden) / Theft / Annoyance

In order create an effective security plan, you must identify your threats. It is important to try to see through their eyes, to gain insight into their mindset and way of thinking. By anticipating how they might act, you can put the appropriate defenses in place. Taking this to the next level, there are even security companies that hire ex hackers and others to test their systems and procedures.

POINT OF ORIGIN

Point of origin identifies whether the threat comes from an authorized user or not. External points of origin are people who do not belong in your database. They may be the stereotypical hacker, just trying to get in and look around, an ex-employee looking for revenge, or even a regular employee getting into a system that they are not authorized to access. The last case can be protected against by strict enforcement of security policy. Make users logout of systems or have password protected screensavers that activate after only a couple of minutes of inactivity should the user leave his or her desk. These policies must be accompanied by harsh penalties. The greater the risk to the company, the greater the penalty, up to and including termination.

Threats with an internal point of origin are those by people who are authorized by the company to access the system, yet misuse or cause damage while doing so. Internal threats are often comprised of accidents or people accessing data for purposes other than what the company intends for them to do. Policies, training, and a comprehensive review of who needs to access what data can go a long way to shoring up the defenses here. Making sure all users are provided with the appropriate training can reduce "accidental" errors, such as over deleting or over updating through the lack of or improper where clauses. Resource limits can also play a role in reducing the negative effects of poor user queries. The DBA must work to limit access to the data so users can perform only the necessary action on the proper

data that they need to perform their jobs. This may include having more powerful user accounts to perform certain tasks. For example, one group of permissions for users entering data and another more privileged account or supervisor account which can delete orders in the event a mistake is made and committed.

MOTIVE

The second attribute of a threat is motive or intent. In many respects, once the damage is done, the motive, whether intentional or accidental, is a moot point. The real purpose of stating the motive is so that we can fully identify all potential threats and damage to our data or impact the performance of our database. Someone accidentally updating all rows in a column to an incorrect value can be just as damaging and cause just as much of a loss or outage as someone who breaks in and alters some data values in order to damage the database. By differentiating threats by motive, we can formulate a plan to defend not only against those wishing to do damage or steal data, but we can help prevent the authorized users from doing things which can be equally damaging. Threats with either motive carry the same capacity for damage, yet are defended against with vastly different strategies. Based solely on the motive attribute, the best defense against intentional threats is to limit access to the data, limit system resources, and to audit and track changes. While these defenses also help protect the data from accidental threats, education and training are imperative to protecting the data. Users get into trouble when they think they know how to do something, but do not have a complete understanding of the commands they are issuing. The undisputed, single most common error of this type is the improperly or unbounded update or delete statement. These are updates or deletes with a where clause that does not act on the proper data set (the data set that the user expects it to act on or simply lack a where clause of any kind. Even experienced users or developers occasionally issue these damaging queries. This is one reason it is so important to limit the ability to issue ad hoc queries.

FORM OF THREAT

The third attribute of a threat is the form in which the threat manifests itself. Security threats come in 4 basic forms: Destructive, Sabotage, Theft, and Annoyance. Each form takes a slightly different strategy to defend against. The potential damage of each type varies from organization to organization and depends on what data is affected. While destruction of data at organization A could cause down time and damage, theft or misuse could be equally or more damaging to organization B. Each form can have different levels of severity and different consequences for each organization and is heavily dependent on which data is compromised, to what extent, and in what physical manner.

DESTRUCTIVE THREATS

The destructive form is where data, files, or objects are deleted or altered extensively. While we are talking about severe damage, it is usually obvious. Applications tend to blow up when key tables are missing, or specific records can not be found. Oracle databases come to a screeching halt when files are deleted. More often than not, destructive actions can be recovered from with the aid of a solid backup strategy, usually resulting in only a short outage or downtime period. The best defense against destructive threats is to limit access at the OS level (to protect files) and to limit granting delete, truncate table, or drop table privileges only on the minimum set of tables possible to only those accounts which need it to perform their jobs.

SABOTAGE THREATS

Sabotage can often be the most seriously damaging form of security violation. The fact that changes can be made which are hard to detect and to determine at what point they happened makes these a daunting threat. While deleting all the data from the pricing table will usually be quickly caught, raising or lowering prices by 10% or 20% could go unnoticed for some time. It would be hard to determine how many sales might have been lost if your prices on the *Web* were sabotaged and raised by 20%. How would you deal with all the sales that were made at that artificially high price?

THEFT THREATS

Theft of data is also a major threat to any organization. Depending on the nature of the data and how it is used, the effects can range from minor to causing the demise of an organization. While gathering customer list or demographic data and selling it is clearly illegal, the effects may vary depending on who obtains this information. This information going to a small non-competing business would have less harm than if it went to the organization's main competitor. What would be the effects on employee morale and retention, should someone steal a list of employees and their salaries from a Human Resources system? What if this information were distributed or made accessible to all

employees? What if it was given to a competitor so they could target and hire away key employees? While these examples would cause obvious problems for the organization, they most often would not be catastrophic. But on the other hand, let's say you work for a medical company. What if someone generated a list of patients with particular illnesses, like AIDS, and made this information available? What if they published it on the *Web*? What would the legal consequences be? What if this information was used for blackmail or extortion? These situations and their accompanying litigation could be catastrophic for an organization. What if the data is credit or financial data? As E-commerce efforts become more wide spread, what could be done by the misuse of this kind of data? While much focus is placed on encryption schemes and other techniques to guarantee the security and integrity of the information in transit, how much emphasis is placed on protecting it once it is inside the database? Clearly, theft of data must now be considered in developing a comprehensive security plan.

Annoyance Threats

Annoyance threats are a collection of other threats that do not clearly fall into the other major categories. They are actions or processes, whether legitimate or not, that are negatively impacting the database. The most obvious of these would be an untuned or poorly written database query. Remember that threats are anything that negatively impacts the database, the data, or the ability to access the data. It might even be a UNIX cron job set to run at an undesirable time. These threats can be difficult to track down and prevent. Good change control management and policy can go a long way in reducing these threats. Ensuring that all changes to the database or system, including application code, are well known and thoroughly tested is a good start. Auditing and analyzing all database access (SQL) and eliminating or controlling the ability to write or run ad hoc queries will prove to be a helpful defense mechanism. Keeping the database on a dedicated database server eliminates many potential annoyance threats by providing greater control of the database's physical environment.

Levels of Security

One of the first steps in creating a secure database environment is to perform a self-evaluation of his or her organization's databases and security policies. This must be thorough and honest. It is often beneficial to take the attitude of someone wanting to damage your database. Think what you could do to your systems. What could cause the most damage? What would be the hardest to detect? What information could you gather that could harm the organization or have value to others? It is important to look at your current state. What security do we have in place? How effective is it? What is the goal of our security? How important is the security of our data? What is the organization's stance on security? The following five categories cover the gamut of security from more or less nothing about security to doing everything possible to ensure database security. Which level does your organization fall into?

Level 1- The Open Door System

This is where there is no perceived need for privacy and/or protection of the data. No real security measures are taken. There is a lack of security policies and little if any enforcement of them. While many systems have this level of security, one must question if it is appropriate. How would the organization fare if this database where unavailable for a period of time? What would the impact be if some of the data were inaccurate? Does it contain information that could readily be published in a newspaper without negatively impacting the organization or giving its competitors an advantage?

Level 2 - Don't Ask, Don't Tell

These are organizations that hide from or dodge the security question, like the child hiding from monster under his or her bedcovers. If I cannot see it, then it cannot hurt me. These organizations do not talk about security, do not think about security, and thus do not have a security problem, or so they think. It might be that they are just too busy to worry about it. Like insurance, this is fine till something happens and they are not protected. There is never extra time to implement security, and it is not an issue that can wait.

Level 3 - The Grand Illusion

The organization believes that its databases are secure. They typically cover the basics, the obvious, and the common industry standards. Most often they are secure against the most obvious threat – the external/intentional/destructive form. The problem here lies in that they believe they are completely secure already, but in fact they are vulnerable to most internal threats. Often the organization's security is implemented inconsistently.

Level 4 - The Locked Door

These organizations understand security and are managing their level of risk. They have covered all the basic areas and are willing to accept a predetermined level of risk rather than do what is necessary to move to the next level of security. This level is typified by generally solid security. They have a security plan and take security seriously. These organizations usually have good database, system, and application security, but often do not go as far as encrypting their network traffic.

Level 5 - Top Secret

These are the secure and tight systems. Typically, they are found in government and military related systems that have a smaller user community. Questions about security, in all its forms, are always included in discussions about the database or in making changes to it. All aspects of the system are considered, including the database and user accounts, the system, and the network.

Types of Security

Application Security

Application security deals with protecting the data from authorized user accounts. Applications provide the users with a controlled view of the data and limit the types of operations that can be performed on that data. This provides some security to the data, but still leaves the database vulnerable to potentially hazardous threats. Everyone hails how great open systems are and the power of ODBC. In terms of security, they open a virtual Pandora's box of problems. ODBC, and in general all 3rd party applications, circumvent the application's limiting effects on the data and operations, thus leaving only the bare database level permissions and grants to protect the data. There are two basic methods of protecting the data from 3rd party applications, they are application password encryption and session enabled roles with passwords. These two methods go a long way to ensuring that data access is controlled only by the application. These 3rd party applications pose one of the greatest internal and unintentional threats to the database. Power users and even developers, without full understanding of the database and its design, often think they can go around the application and alter or add data faster and easier. Fast and easy is not always the best way though. Most often they are not skill or informed enough to write proper and efficient queries to perform the intended task. Tools that use ODBC or generate their own SQL are notorious for issuing poor SQL. I worked on a system where a power desktop user thought it was easier to put data into an MS ACCESS database, use ODBC to link his tables, and insert or update data. Often this was on a major intersection table. This almost always results in a

table lock. It might have gone unnoticed on a system with 5 or 10 users, but on an OLTP system with 600+ users, people scream in a big hurry.

Another reason to control access through SQLPLUS type tools is that it allows access to some data dictionary views that can compromise security. As innocent as the all_users view may seem, it can allow users to find potential holes in your defenses by giving the names of accounts which the DBA may not have protected. The section on Account Password Security mentions some of these accounts. Not that a knowledgeable threat would not already try these accounts, which you hopefully have accounted for in your database security plan, but it is just another example of how seemingly benign information can be used against the database.

Application Password Encryption

What is Application Password Encryption? It is a program module, part of the main application program, which according to a predetermined algorithm, manipulates the password before sending it to the database for verification and acceptance. Simply put, the user enters his password, but a different password is sent to the database for verification. This eliminates any access to the database through any means other than the application, unless you know the encryption algorithm. This stops all 3rd party software access to the database, including connecting from SQLPLUS, ODBC, or any other tools that log into the database. The key here is to protect the encryption algorithm. Of course, whoever writes and compiles the password module will know the encryption algorithm. The DBAs will need to know the algorithm. Prudence will have someone in management be in on the algorithm selecting process, in addition to seeing that the algorithm is properly documented somewhere. This scheme is not foolproof. It only works as well as the encryption algorithm is kept secret. This method works well with large numbers of users. The algorithm can be changed periodically or as needed. This is not trivial, but can be accomplished by creating scripts that reset each user's password to their user id modified by the new encryption algorithm. Another very important part of this strategy is to provide a password maintenance unit or module in the program. This allows the user to change the password and have the encryption scheme applied to their new password before submitting the password to the database to actually be changed.

So, what are some of these encryption algorithms? They are not rocket science. Some examples of these algorithms include:

```
Insert a special, non-reserved character or number between the 3rd and 4th characters
johnson becomes joh&nson or joh5nson
Reverse the order of the characters
johnson becomes nosnhoj
Reverse the order and insert a special character
      Johnson becomes nos&nhoj
Repeat the character in one or more positions:
      Johnson becomes jjoohhnnssoonn or johhnnson
```

All these can be accomplished in almost any programming language by using simple string manipulation functions with minimal effort. For those wishing for or requiring a more intricate algorithm, the sky is the limit. Other possibilities include creating a mapping array in the encryption module, which has each possible letter or number mapping to another letter. This may be by position, such as +7 where a maps to h, b maps to i, c maps to j, d maps to k, etc. This can be done incrementally by position. The first position is +1, the second character is +2, then +3, etc. The mapping can also be completely random and stored in a static array inside the encryption module. To further complicate the encryption scheme, the mapping can be combined with any of the string manipulation algorithms. These schemes are all within the abilities of the average programmer to implement. It is important to remember the goal of this form of protection. It does not guarantee protection from serious, hard core hackers, but if the algorithm is kept secret, it will provide a significantly higher level of security and force access to the database to occur through the intended application and not via ODBC or the host of other 3rd party tools. The possibilities are virtually limitless.

Session Enabled Roles with Passwords

Another method of protecting access to the database is to use session enabled roles with passwords. This is done by granting all the necessary grants and permissions for a user or type of user to a role. Many organizations already do this, but they typically create the user giving them this role as the default. Here we are talking about having the user,

without the role, invoking the application and having the application grant the role to the user. This is done in the application by issuing the set role command (SET ROLE role IDENTIFIED BY password). This gives the user the role for the duration of their current session. Once they log out, the user account loses the role. Thus the only way to get access to the tables is through the application. This allows the developers or others who know what the role's password is to still connect through other means, such as 3rd party tools. It also allows the user to have only the permissions or grants they need in that particular application. A user might be using two applications, possibly concurrently, and each session would have only the permissions for that application.

ACCOUNT PASSWORD SECURITY

Account password security involves keeping each account in the database secure. The first rule is, wherever and whenever possible, avoid the use of generic or shared accounts. They are trouble from the word go. Yes, they are easier to implement and manage than hundreds or more accounts, but the easy way is rarely the best way. Having generic or shared accounts is the single best way to invite trouble into your database.

The second rule is to know what each account is for and who uses it. Actively manage your accounts. One of the best ways is to get a process in place that puts the DBA in the loop with the Human Resources Department. See that you are notified of every person who quits or is terminated. See that their logins are disabled (change the password) or removed immediately upon their last day, if not sooner. If you are using the OS Authentication option, thus having the user login to the server and letting the OS handle the password verification, see that the system administrators are aware of a person leaving the company. Just because you remove them from the database does not mean that they no longer pose a threat to your database. (See the next section on System Security).

Now let us talk about some of the biggest account/password violators. These are common accounts which are often are very powerful. These accounts include oracle, sys, system, scott, dbsnmp, and context.

In UNIX, the oracle account is the core system account. It owns virtually everything at the OS level. The account must be initially created by someone with "root" access. This is rarely the DBA and almost always the system administrator. The number one most common and most obvious password for the system administrator to give this account is oracle. The second is the company's name. The issue is that in a number of cases, all far too often, this does not get changed. Even if you do initially change it, how often is the password changed? Is it changed every time a DBA leaves your company? Usually this is not the case. If you are actively managing this account's password, great, if not, there is no time like the present to begin.

Next in importance come the sys & system accounts in the database. Access to these accounts give a person ultimate power in the database. We are all quite aware that they come with well-known default passwords. We are all quite aware we should change them immediately after creating the database. Unfortunately, for some, they just do not get around to it. Major problems here. There is no excuse for not resetting both the sys & system account passwords, and the oracle account password. Stop here. If you can honestly say that you are 100% sure that all your databases have these passwords changed, then go on, if not go out and check and/or change them, then come back and finish reading this paper. If you have changed them, but have not done so in a year or more, or have not done so since the last DBA left your organization, make a note and begin planning it immediately after finishing this paper. Perhaps Oracle Corporation will make a slight revision in the server software so that in the future the sys and system account passwords will be in the form of a user defined parameter at the time of database creation.

Perhaps the most famous Oracle database user is everyone's friend Scott. Hard as it may be to believe, there are databases out there with the scott/tiger account/password. Usually, these were set up so that someone could redo some things they learned in a class to practice or show others. Even with limited privileges, this is not a good idea.

Does dbsnmp sound familiar? Think you have seen it somewhere? It certainly sounds cryptic and important. Any idea what this account does? Well, it shows up as part of the standard install in Oracle versions 7.3.3 and beyond. It is created for the Intelligent Agent for Oracle's Enterprise Manager. The unintelligent part is the password. Care to guess? Yes, it is the same, dbsnmp. Ouch! If you are not using OEM, get rid of this login, if you do use OEM, then change the password for this account. The documentation is not easily found, but Oracle support can provide you with the procedure. Why is this important? Dbsnmp creates table and its default tablespace system. This account can fragment or fill up your system tablespace and cause all kinds of havoc in your database.

A similar situation exists when the CONTEXT cartridge is installed. Installing this option creates a user account ctxsys with a password of – you guessed it ctxsys! This is a default account that is automatically created. Again, if you have this installed, you probably need it, but get the password changed.

Why are these accounts singled out? What is it that makes these accounts so important? It is that they possess very powerful grants and privileges. Think twice before grant more than connect and create session to a user account. Never give out the DBA role. Be extremely careful before granting anything with the admin option. It is easy to give out extra grants and permissions when needed, but it is often difficult, and sometimes too late, when you need to revoke them.

SYSTEM SECURITY

System level security is concerned with the area around the database. Consider it the perimeter defense for your database. Regardless of how well you protect the data inside your database, encrypt, and force good password discipline, you are still vulnerable unless the environment in which your database lives is well protected. As a DBA, you do not usually have direct control over this security, but it is important you raise the issue and do what you can to make it more secure. What can happen if someone can gain access to the server where your database lives - delete or alter SQL scripts, delete data files, filling up file systems? The effects of such actions run from mild inconveniences to serious outages. What can a DBA do to protect his or her databases from this? The first step is to talk with the system administrator. Discuss the security of the server. Learn what other application reside on the server, who has logins to the server, and what the system administrator is doing to ensure the security of the server. If at all possible, see that your databases are on dedicated database servers, such that there are no other applications or logins at the OS level into the box.

Make sure the system administrators are aware of the security needs and take it seriously. You think some strange breach could not happen to you, under your watch? Once I was DBA on a system prior to working at Sprint, administrating a production database of 15 GB and 600+ concurrent users. Someone else in the company had come across a script on the internet to try to break some 128-bit encryption code. It was some kind of contest thing. He asked one of the system administrators if he could have an account on one of the large DEC Alphas, and they gave him one of the guest accounts. I received calls that the system was running sluggishly. I checked all of the normal database items and could find no problems. To complicate matters, we occasionally would have network related issues and the application was not as well written as I would have liked. Finally, I widen my search and am looking at a full list of OS processes (ps –ef) and I find this process owned by guest that is sucking up CPU like crazy. It had been running since the night before! It was the guy logged into this guest account running some script off the internet, of which he had no real idea what it did, on the same system as my production database. To say that I was upset would be an understatement. Fortunately the script was benign and did no damage to the system. The company still suffered the lost productivity of myself, at least one other DBA, and the 600 or so users who were trying to use the very slow reacting system. If this can happen to me, it can happen to anyone else.

Another aspect of system security is the network itself. Threats can gain physical or logical access to your system with software that is readily available commercially or on various bulletin boards. They can reroute network packets through another workstation where packets are viewed and can be modified or deleted. Oracle, since version 7.1, has encrypted passwords at login, but does not encrypt them when changing the password. Oracle8 has provided the Advanced Networking Option (ANO) in response to this threat. For data encryption ANO uses the Data Encryption Standard, DES, with key lengths of 40 and 56 bits and the RSA RC4 algorithm from RSA Data Security Inc. with key lengths of 40, 56, and 128 bits. Oracle's ANO also uses cryptographic checksumming. This is where the value of the packet is run through a special hashing algorithm, MD5. Protecting your system from these threats is much more intricate and each system must be evaluated as to whether this type of protection is necessary for your system.

ORACLE8 SECURITY FEATURES

Oracle8 provides a number of features to enhance database security. Most important among these features are robust password management options and the Advanced Networking Option. Here we will examine the various password management options. A brief description of the Advanced Networking option can be found in the section on System Security.

Oracle7 provided only minimal password management. Basically, if granted permission, the user could change his or her password. This is provided that the user has access to a SQL prompt or the application has a password management module. There was no way to enforce any rules regarding the make up or expiration of passwords. Much like many products and operation systems, such as UNIX, parameters can be set for account locking, password complexity verification, password history, and expiration periods. A brief description of each follows:

- **Account Locking** –A user account can be locked out either manually or after a predetermined number of failed login attempts.
- **Password Complexity Verification** – Allows the DBA to specify certain characteristics or parameters for which all passwords must comply. These include minimum or maximum length or that they must contain at least a certain number of numeric places.
- **Password History** – Oracle can check to see that a new password is not being reused either for a specific length of time or for a set number of password changes.
- **Password Expiration** – This parameter allows the DBA to specify the interval before the database requires the user to change their password. It allows for a grace period for the user to change their password. After the grace period, the user's account may be locked out of the system.

These parameters may be managed via profiles. Restriction on system and object resources may also be maintained in the profiles. Some of the restrictions on system and object resources include:

- SESSION_PER_USER
- CPU_PER_SSESSION
- CPU_PER_CALL
- CONNECT_TIME
- PRIVATE_SGA
- IDLE_TIME
- LOGICAL_READS_PER_SESSION
- LOGICAL_READS_PER_CALL
- COMPOSITE_LIMIT

By employing these restrictions via profiles, you can limit the amount of resources (CPU, I/O, etc.) that a user account can consume during a session or for a single query. A robust security plan can incorporate these limits in profiles in addition to roles to limit to help protect the database.

CREATING A SECURITY PLAN

With a solid understanding of what security is and what all the possible threats to your databases are, including their consequences, it becomes time to put together a comprehensive security plan. Identifying the threats and developing the plan are the hard part. Once it is in hand, selling it to management should be easy. Fear is an incredible motivator. Simply pose some of the threat scenarios you have identified and ask management what the consequences of such intrusions might include. With the litigious nature of today's society, receiving management's backing for your plan is virtually assured.

A good security plan consists of a comprehensive set of plans, strategies, and policies targeted at protecting the organization's data. It is the result of a group effort that includes DBAs, system administrators, application team leads, business unit leads, and upper management. The plan is reviewed and updated on a regular basis. It begins by identifying what all the possible threats and the possible consequences of each might include. Plans are then made in each of the areas discussed in this paper, application, account/password, and system security, to prevent the identified threats. The next step is create formal policies to help tie all the security efforts together, making them work in concert to form a protective shield. Some examples of these policies are:

- Procedures to remove logins of exiting employees on or before the day they leave.
- Policies against sharing passwords or using another person's account.
- Leaving your computer logged in while being away from your desk (or without a password-protected screensaver).
- Copying production data into other environments, i.e. test or development databases.

- Altering production code, implementing untested code, or running unauthorized queries against a production system.

Once these procedures have been established, they must be communicated to the whole organization. Security must be made part of the organization's culture. The whole organization must think security. It needs to be recognized, as an integral part of every project, not as something that slows down the process, is extra work, or as something that can be reduced or eliminated when facing deadlines. The security plan must have management's complete backing. This backing must include giving these policies teeth and enforcing them. Breaches of these policies must be considered actionable offenses, with appropriate penalties. Depending on the data and the organization's need for a secure environment, the consequences can range from documenting the offense and being "written up" to and including termination. Termination may seem harsh, but one must remember that some of these offenses could have serious financial consequences for the organization.

SUMMARY

The key to database security and security in general is knowledge and awareness. You must be aware that real threats to your data and system exist. You must have the knowledge to identify the threats, analyze the likelihood and potential impact of each threat. Now you must do an evaluation of your current security efforts to determine what areas of your database system are vulnerable and then create a comprehensive strategy to protect them against all threats you have identified. Armed with this plan, you must get management's complete backing in order to effectively implement the plan. The plan must be a living and evolving strategy. It needs to be reviewed, tested, and updated on a regular basis. Failure to implement a complete and comprehensive plan will certainly reduce your risk, but may create a false sense of security. Security needs to be discussed early on in the planning phase for projects. Discuss the security needs of the project, the application, and the data, while looking at how to apply the existing security plan to this new project. Each organization is unique and has its own security needs. No system is 100% percent secure. DBAs need a great deal of access to the database in order to administrate it. System administrators have great power at their hands also. Someone will know your encryption schemes and other key facts. The goal here is to put highly qualified ethical people in these positions, minimize the chance of harm to your databases and systems, and limit the consequences of these threats. It is never too late to start thinking about.