Best Practices to Protect Databases behind RDBMS-Powered Web Applications

Ahmad Hammoud and Ramzi A. Haraty
Lebanese American University
Beirut, Lebanon
Email: {ahammoud@gu.edu.lb, rharaty@lau.edu.lb}

Abstract

This article focuses on the best practices necessary to protect the databases behind the Web applications. Web masters, database designers, databases administrators, and database developers should be trained to follow these practices in order to minimize unauthorized database access. Several topics will be discussed, including: access through Web server, indirect access to tables, transaction log versus detailed action log, trash database, files access control, sensitive information changes, and user's information.

Keywords: Database security and web applications.

1. Introduction

This article focuses on the best practices necessary to protect the databases behind the Web applications. Web masters, database designers, databases administrators, and database developers should be trained to follow these practices in order to minimize unauthorized database access. Several topics will be discussed, including: access through Web server, indirect access to tables, transaction log versus detailed action log, trash database, files access control, sensitive information changes, and user's information. This work applies to all Database Management Systems (DBMSs) that are currently in use. This includes Oracle Database Server, Microsoft SQL Server, and IBM DB2. The following table lists some RDBMS along with the host platform [4].

DBMS Vendors	DBMS version addressed	Host platform
Oracle	Database Server 8i, 9i, 10g	Windows, UNIX, OS/390
Microsoft	SQL Server 7 and 8 (2000)	Windows
IBM	DB2 Universal Database 8.1	Windows, UNIX

The remainder of the paper is organized as follows: Section 2 presents the background of this work. Section 3 presents the database best practices. And section 4 contains the conclusion.

2. Background

More and more frequently, malicious users are attacking data stored within a DBMS. The effect of such an attack may result in unauthorized access to sensitive data. DBMSs have also joined the ranks of malicious attacks victims [3].

Before one can handle database security, it is of crucial importance to understand database concepts. Businesses cannot afford the risk of an unauthorized user changing or even observing the data in their databases [1].

There are three types of concerns related to the security of databases. These types are: "incorrect data modification, unauthorized data observation, and data unavailability" [2]. Unauthorized data observation occurs when database users access information that they are not authorized to view. Incorrect data modification is either intentional or unintentional. Data unavailability takes place when "information crucial for the proper functioning of the organization is not readily available when needed [2].

Despite a rigorous set of security criteria, vulnerabilities and attacks still occur even in the most advanced systems. In February of 2003, a breach in the security of a database owned by DPI (Data Processors International), a credit card processing company, released about eight million credit card numbers to attackers. This was estimated to cost the credit card companies about $200 million dollars (cancellation and renewal expenses) [6].

The 2002 Computer Crime and Security Survey of the Computer Security Institute revealed that every year, more than half of all databases have a kind of breach. It also revealed that the average breach amounts to nearly $4 million in losses [7].

By applying security techniques without understanding them, it is possible to compromise security [1]. This article is meant to help those in charge of database security to minimize the risk of both unauthorized data observation and modification.

Keishi Tajima of Kyoto University states that "User access to a database is either an action to get some information from the database, or an action to give some information to the database in order to make it reflected by the database state" [8]. With this in mind, we find that database users share a great deal of responsibility.

The most common threats to the data stored inside your database include: [5]

- Unintentional threats from either known parties or accidents. Authorized users will inadvertently make some mistakes. They may also see data they should not see. Sometimes, they may delete or change records that they should not have access to.
- Intentional threats from known parties such as hackers who will benefit from accessing or intentionally damaging data that they should not see.
- Threats from anonymous parties or uninvited intruders. These are mostly Internet-based threats from intruders with anonymous access.

The following sections provide a list of best practices that could possibly enhance the database security. All those in charge of the database security should be encouraged to follow these practices. The list presented in this article is not exhaustive; there are many more practices to follow. The aim here is to raise the level of awareness without the use of any extra tool.

3. Main Thrust: Database Best Practices

Access through Web Server

It is a big problem to have your database server exposed to the Web without any type of protection. How can you guarantee that it will not be attacked? Who can make sure it will be accessed only through your Web server? Obviously there is no guarantee to this unless certain measures are taken. This paper intends to provide possible solutions to this problem.

Can the Access to the Database Server be Limited to the Web Server?

The answer is yes. There are many ways to achieve this. Some IT experts may recommend the use of ready-made packages. Although some of these packages are trustworthy, some do not guarantee a complete access control. The approach suggested here will make certain that the databases are protected without using any extra tools. The suggested approach is efficient because it makes use of triggers to control access to the database. The advantage of the approach lies in its ability to run on the database level. Before giving any illustrations, the trigger concept will be defined to help explain how it will contribute in the solution.

What Is A Trigger?

Triggers are considered a special type of stored procedure. They automatically run when a DML event occurs in the database. This is to say that they are invoked whenever an INSERT, UPDATE, or DELETE statement is executed. Trigger can include complex SQL statements. Both the statement firing the trigger and the trigger itself are considered a single transaction. This means that it can be rolled back if a severe error occurred. When this happens, the whole transaction will automatically roll back, which implies that all changes made will not be saved to the database.

Proposed Solution

Triggers can detect incorrect DML operations and malicious transactions. They can be designed to take proper action when such an activity is detected. They can also be used to enforce some restrictions, which are more complicated than the ones that can be defined using the CHECK constraint. Triggers can check the state of the corresponding table before and after DML operations and take proper actions accordingly.

Typically, all tables are guarded by:

1. An UPDATE trigger that fires when an UPDATE statement is invoked to change the content of the table.
2. A DELETE trigger that fires when a DELETE statement is invoked to delete records from the table
3. An INSERT trigger that fires when an INSERT statement is invoked to add new record(s) to the table.

Taking advantage of the above guards, the database administrator may design three triggers for each table in order to check whether the user accessing the table is the Web server's account. The following code snippet shows the syntax that might be used to achieve this goal. It is the code necessary to create an INSERT trigger that will make sure that the user trying to add a new record is the Web server. If this is not the case, the whole operation will be rolled back and an entry is made in a log table that will be reviewed by the system administrator. Here is the trigger:

```
CREATE TRIGGER
Employees_ITrig_OnlyIIS ON
dbo.Employees
        FOR INSERT
AS
BEGIN
        If user <> 'IUSR_WebServerName'
                BEGIN
                        Rollback
                        Insert Into
MaliciousTrialsTbl (Operation, TrialDate,
Usr)
                        VALUES
('Update PersonTbl',getdate(), user)
                END
END
```

The user name 'IUSR_WebServerName' can be, for example, replaced by the username that ASP.NET uses. Similar triggers can be defined for INSERT and DELETE operations. If this approach is followed, the database administrator can be 100 percent sure that "his/her" database is never touched but by the Web server. Another important point that needs to be emphasized here is the ability to control the reaction. If an undesired operation is about to occur, you, database administrator may decide to perform any or all of the following:

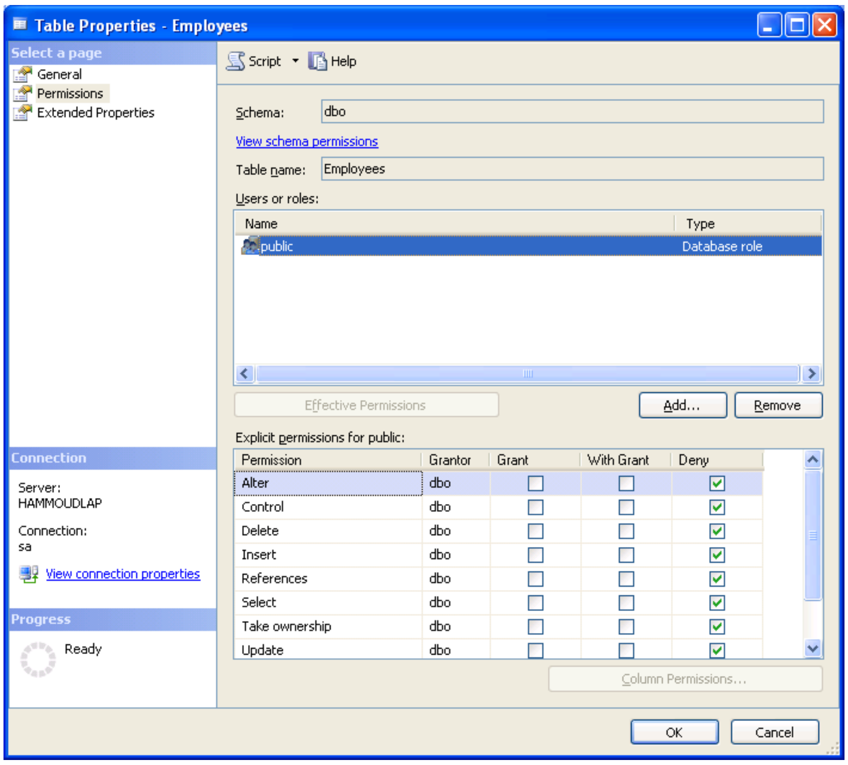
- Rollback the whole transaction
- Allow but notify
- Make an entry in the log file
- Send an email
- Call a stored procedure
- Any other custom action

This will add certain value to the "safety" of the database by enhancing its security level. It is an inexpensive, easy-to-implement, efficient, proved, and simple way to enforce access control, one of the worst nightmares of the Web masters, database designers, databases administrators, and Database developers.

Indirect Access to Tables

Whatever the case is, direct access privilege should never be granted to any user other than the database owner. What is meant by direct access privilege is the permission to access the table directly and not through a stored procedure. The user should not be allowed to open a table, delete records from it, or update records. Instead, all the users should be given permission to run selected stored procedures each of which performs a well defined task. If the user is given permission to access the table immediately, then the control is minimal. The best practice is to deny all users access to all tables. Then, the developer, or the database administrator, determines which stored procedure is needed by which role (group of users). Every role should be given only the permission to execute the needed stored procedures; neither more nor less.

Figure 1 shows a snapshot obtained from the SQL Server Management Studio. It shows the table properties of the Employees table. Note that the Public group, to which all users belong, is not allowed to access the table. All types of permission are denied. These types are: Alter, Control, Delete, Insert, References, Select, Take Ownership, Update, and View definition.



Schema: dbo

Table name: Employees

Users or roles:

Name	Type
public	Database role

Server: HAMMOUDLAP

Connection: sa

Explicit permissions for public:

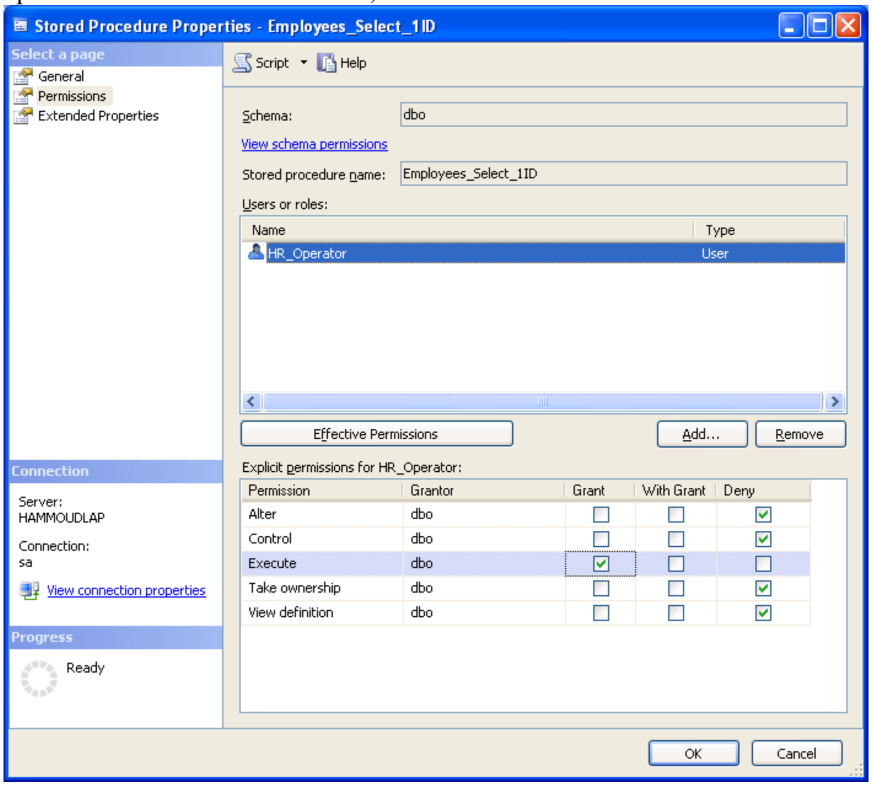
Permission	Grantor	Grant	With Grant	Deny
Alter	dbo	☐	☐	☑
Control	dbo	☐	☐	☑
Delete	dbo	☐	☐	☑
Insert	dbo	☐	☐	☑
References	dbo	☐	☐	☑
Select	dbo	☐	☐	☑
Take ownership	dbo	☐	☐	☑
Update	dbo	☐	☐	☑

Figure 1: Deny Access to Public Role

Hence, how are the users expected to communicate with the database if they are not allowed to access any table? To be able to answer this question the concept of Least Privilege needs to be defined. It is to give users only the least security privileges needed to complete a given task. While this is in mind, the Web developer, the database administrator, or/and the system administrator gives the roles (groups of users) only the minimum rights required. For example, the users who should be able to see the information of a certain employee should be given the permission to execute the following stored procedure:

```
CREATE PROCEDURE Employees_Select_1ID
        @ID int
AS
BEGIN
        SELECT * FROM Employees WHERE ID =
@ID
END
```

Obviously, the users will be able to SELECT only 1 row through this procedure. To be on the safe side, the procedure may only specify 1 or 2 columns instead of using the star (*) which means to return all the columns. Figure 2 shows the permission granted to the HR_Operator, which should be allowed to EXECUTE the above stored procedure. Following the concept of least privilege, the user (or the group of users) is given "Execute" but everything else is denied. As seen in Figure 2, the HR_Operator cannot do any of the following: Alter, Control, Take ownership, and View definition. Although he/she is granted the EXECUTE, he/she will not be able to grant others. This is what least privilege is about.


Stored Procedure Properties - Employees_Select_1ID

Select a page
- General
- Permissions
- Extended Properties

Script Help

Schema: dbo

View schema permissions

Stored procedure name: Employees_Select_1ID

Users or roles:

Name	Type
HR_Operator	User

Effective Permissions Add... Remove

Explicit permissions for HR_Operator:

Permission	Grantor	Grant	With Grant	Deny
Alter	dbo	☐	☐	✓
Control	dbo	☐	☐	✓
Execute	dbo	✓	☐	☐
Take ownership	dbo	☐	☐	✓
View definition	dbo	☐	☐	✓

Connection

Server:
HAMMOUDLAP

Connection:
sa

View connection properties

Progress

Ready

OK Cancel

Figure 2: Stored procedure – least permissions

Bear in mind that the user can still perform SELECT, UPDATE, DELETE and other data manipulation statements although he/she does not have the permission to access the table as long as he/she is granted the permission to execute a stored procedure that does so. Thus, the typical secure situation in this context is when all the users are given the permission to execute stored procedures. No one, except the database owner, should be able to access the tables directly.

Transaction Log vs. Detailed Action Log

Surveillance cameras record events 24 by 7. They keep recording everything that is captured. You refer to them only when there is something wrong in order to discover who did what. However, a question might be posed here: What can be done so that the camera does not record everything? There is a need to let the camera record the 24 hours as a list of several events so when the video tape is reviewed, one can check it in a matter of seconds instead of watching the whole collection of video tapes. For example, suppose you are responsible for 10 surveillance cameras in a mall and a burglary has occurred. The time at which it took place is not known but most probably it has happened during the previous week. So you are supposed to watch 10 x 24 x 7 = 1680 video tapes. For more than one week or more than 10 cameras, the case is worse. What can a human being discover if he/she is supposed to watch 1680 hours of recorded actions! The answer is not that much. Hence, another strategy should be followed.

The same concept applies when dealing with applications. Suppose a penetration occurred and you wanted to investigate the case. You know that it happened a week earlier. You will open the log file and you will have to review hundreds of entries. To facilitate your mission, an alternative approach should be followed. Instead of recording every single hit and access to objects which will result in huge log files, use only one entry to record the transaction. This can be achieved through the stored procedures. A stored procedure may call other several procedures. If every action is to be recorded, the single transaction may be recorded in the form of multiple entries. Instead, a chain of inter-related activities should be recorded using a single entry. Consider the following stored procedure:

```
CREATE PROCEDURE Customer_Delete_1ID
        @ID int
AS
BEGIN
    1.  IF dbo.HasPendingOrders(@ID) = 1 GOTO
        HasPendingOrders
    2.  IF dbo.DueFeesGreaterThan0(@ID) = 1
        GOTO DueFeesGreaterThan0
    3.  DELETE FROM CustomersOrders WHERE
        ID = @ID
    4.  DELETE FROM CustomersHistory WHERE
        ID = @ID
    5.  DELETE FROM CustomersContactInfo
        WHERE ID = @ID
    6.  DELETE FROM Customers WHERE ID =
        @ID
    7.  MakeEntryToTheLog
        'Customer_Delete_1ID', @ID, 'Succeeded'
        GOTO Done
        HasPendingOrders:
        MakeEntryToTheLog
        'Customer_Delete_1ID', @ID, 'Canceled:
        HasPendingOrders'
        GOTO Done
        DueFeesGreaterThan0:
        MakeEntryToTheLog
    'Customer_Delete_1ID', @ID, 'Canceled:
    DueFees > 0'
    Done:
END
```

Clearly, seven entries to the log file will be made if every action is to be recorded. Nevertheless, only one entry should be recorded as a result of executing the above stored procedure. This way, the system administrator will choose the transactions to be monitored and will add lines similar to the seventh one in the above stored procedure.

Trash Database

The idea of trash database is very simple: before updating data or throwing it away, save the original copy in a database that will be cleaned on a scheduled basis. This database will be used only in case there is a need to do so. If everything goes well, it will remain intact. It has the same structure as the database in use. Although saving a copy of the data before it is updated or deleted will slow down the performance, it may be of crucial importance if you want to:

- know who did what,
- know previous versions of a certain record, or
- restore a previous state of a certain table.

Sometimes, the database administrator may decide not to apply this to every table in the database. Only several important tables may be chosen to be treated as such. The following DELETE trigger will save a copy of the original rows in a trash database before executing the delete statement:

```
CREATE TRIGGER Employees_DTrig ON
Employees FOR DELETE
AS
BEGIN
        INSERT INTO TrashDB..Employees (ID,
EmpName)
        Select ID, EmpName from [deleted]
END
```

Files Access Control

Suppose you are a developer writing a Web application that allows users to download files. The user should be able to download only his/her files. How will you implement this?

You may put all of these files on the hard disk of the Web server inside a folder and show the user a hyperlink when he/she logs in. The download process will start when the user will click the specified hyperlink. This sounds good but what if the user immediately types a URL in the address bar to download a file that does not belong to him/her. He/She will be able to do so. What can be done to prevent the user from accessing files that do not belong to him/her?

If, for instance, every user is using his Windows account, then 'who can access what' can be controlled by setting permissions (Access Control Lists or ACLs) on all the resources. But if the Web server is configured to use the IUSR_machinename account, then this approach is useless because all resources will be accessed using the same account. So what is the alternative?

The following steps should be followed:

1. For each user, create a subfolder within the parent directory.
2. Let the name of each subfolder be 25 characters (alphanumeric).
3. Save the pair <Username, Subfolder> to the database.
4. Configure your Web server to prevent directory browsing so that users will not be able to know the subfolders within the parent directory.

Now, when you want to show the user his/her hyperlink, you will contact the database, retrieve the pair <Username, Subfolder>, and create the hyperlink accordingly. Because directory browsing is not allowed, the user will not be able to guess the folder of the others. It is an efficient way to control access to files. You need to be very cautious. The name of this folder should never be shown. Although it is a 25-character name, some hackers standing behind the user's shoulder may be able to write down this name and use it later. This is called shoulder surfing. A popup window, for example, does not show the URL. The same also applies if your page contains a media player control to play the file.

Sensitive Information Changes

All those events that affect users' account should be recorded. This is extremely important when dealing with sensitive applications such as financial packages. All of the following types of changes should be tracked:

- **Profile changes**
  Changes to personal information should always be recorded. This includes e-mail address, phone number, address, salary, and any piece of information related to the credit card.

- **Password changes**
  Every time the user changes his password, make an entry in your log file. Never allow such an activity without tracking it.

- **Modify other user**
  Every time an administrator changes the profile of any other user, make sure you record the event, the date and time of change, previous values, and the user who performed the change.

- **Add/Delete user**
  Record critical events such as creating new users or deleting old ones.

In these cases, as much detail as possible should be logged.

User's Information

To improve the ability of tracking the users' activities, users' information should be recorded for every row. This implies that every record in every table should include the following values:

- The name of the user who made the entry.
- The date and time of the entry.
- The name of the last user who updated the record.
- The date and time of last update.
- The IP of the user.
- Session ID (sometimes).

How to Do It?

The developer or the database administrator may write triggers to fill these columns, which will be of high importance when it is time to track the entries. The following is a trigger that will automatically run when a record is about to be inserted into the Employees table. Note that it will automatically fill the four fields: AddUsr, AddDate, LastUpdUsr, and LastUpdDate.

```
CREATE TRIGGER Employees_ITrig
        ON  Employees
        FOR INSERT
AS
BEGIN
    Update Employees
    Set AddUsr = user, AddDate=getdate(),
        LastUpdUsr    =user,    LastUpdDate    =
        getdate()
    Where ID in (Select ID from [inserted])
END
```

Another example is the Update trigger of the same table:

```
CREATE TRIGGER Employees_UTrig
        ON  Employees
        FOR UPDATE
AS
BEGIN
    Update Employees
    Set  LastUpdUsr  =  user,  LastUpdDate  =
getdate()
    Where ID in (Select ID from [inserted])
END
```

What Information Does the Browser Send?

Every time a Web site is visited, the user's browser will automatically send several pieces of information to the Web server. All of the following will be sent:

- Host address
- Web browser's version
- Web browser's language
- Files accepted by the Web browser
- Characters accepted by the Web browser
- Browser encoding
- User name
- HTTP port of the computer

In addition, the settings of the computer can be attained if the user enabled JavaScript on the Web browser. This includes:

- JVM, or Java plug-ins
- FTP Password
- Current resolution
- Maximum resolution
- Version
- Color depth

- Platform
- Anti-aliasing fonts

False Assumption/HTTP referrer header
Many Web developers do not fully understand the way HTTP and/or HTML work. They sometimes mistakenly rely on the HTTP referrer header as a security method. Supplied by the client, the referrer header contains the referring page address. Since it is supplied by the client, the referrer header is easy to spoof. For example, try to Telnet to the HTTP port (port 80) as follows:

```
GET / HTTP/1.0
User-Agent: Spoofed-Agent/1.0
Referrer: http://www.lau.edu.lb/spoofed/referer/
```

As you see, a fake user agent header and a fake referrer header are submitted. As a matter of fact, any piece of information submitted by the user can be spoofed, including the user's IP address.

4. Conclusion

Several database practices are very important to ensure the "safety" of the database. Security relies on the concept of multiple defense lines. One should never count on one defense strategy. The more defense lines you have, the more secure you are. Depending on a single line will lead to penetration as soon as it is breached. The practices presented in this chapter include:

- Allowing the Web server to be the only one that can access the database server.
- Restricting direct access to the tables and giving the roles only the minimum rights required to complete their tasks.
- Logging transactions as compared to logging detailed action
- Having a trash database to store original versions of data before being updated.
- Controlling files access control to prevent a user from accessing a file that does not belong to him/her by simply typing the URL of the file in the address bar.
- Recording sensitive information changes such as password.
- Saving user's information to enhance the level of security.
- Storing pieces of information sent by the browser such as host address.
- Warning not to rely on the HTTP referrer header as a method of security.

Table 1 summarizes the practices recommended in this document.

Table 1: Summary of the Practices Followed.

Practice	Details
Access through Web server The triggers can guard against malicious operations. If the user trying to perform an operation is not the Web server (or .NET) account, take action	If an undesired operation is about to occur, you may rollback the whole transaction, allow but notify, make an entry in the log file, send an email, call a stored procedure, and/or any other custom action
Indirect access No user should be able to access the tables	Users should be allowed to execute stored procedures that perform very limited operations. The best practice is to deny all users access to all tables
Transaction log vs. detailed action log	Do not record every user's action. Instead, record the whole transaction that includes many steps as a single entry in the log table
Trash Database Before updating data, keep a copy of the original version in trash DB	Trash DB will be cleaned on a regular basis. Not all tables, only selected ones, should be treated as such
Files Access Control To prevent a user from accessing a file that does not belong to him by simply typing the URL of the file in the address bar, create a subfolder for each user and save the pair <Username, Subfolder> to the database	Because directory browsing is not allowed, the user will not be able to guess the folder of the others The Web server should prevent directory browsing Avoid showing the name of this folder to prevent shoulder surfing
Track sensitive information changes	This includes: profile changes, password changes, modify other user, and add/delete user.

Practice	Details
User’s Information Every record in every table should include the user who made the entry, entry date and time, last user who updated the record, last update date and time, and user’s IP	The developer or the database administrator may write triggers to fill these columns which will be of high importance when it is time to track the entries.
Save the info sent by the browser to the database.	This includes: host address, Web browser’s version, Web browser’s language, files the Web browser accepts, characters your Web browser accepts, browser encoding, user name, and Http port of the computer.
Never rely on the HTTP referrer header as a method of security	It is too trivial to spoof

References

[1] Aaron, Nathan. Oracle Database Security. Retrieved September 22, 2006, from: http://www.infosecwriters.com/text_resources/pdf/Oracle_NAaron.pdf.

[2] Bertino, E., Sandhu, R.. “Database Security - Concepts, Approaches, and Challenges.” Dependable and Secure Computing, IEEE Transactions on Volume 2, Issue 1, Jan.-March 2005 Page(s):2-19.

[3] Castano, S., Grazia Fugini, M., Martella, G., Samarati, P. Database Security. New York: ACM Press/Addison-Wesley Publishing Co. 1995.

[4] DISA Database Security Technical Implementation Guide Version 7, Release 2. (Developed by Defense Information Systems Agency for the Department of Defense). Retrieved October 01, 2006, from: http://iase.disa.mil/stigs/stig/database-stig-v7r1.pdf.

[5] FileMaker, Inc. FileMaker 7, Security Guide. Retrieved October 1, 2006, from: http://www.filemaker.com/downloads/pdf/fm7_security_guide.pdf.

[6] Giuliani, M., Lobner, E., & Wagner P. Investigating Database Security in a Networked Environment. Retrieved September 17, 2006, from: http://www.cs.uwec.edu/~wagnerpj/mics2006/DatabaseSecurity.pdf.

[7] Nevins, Scott. Database Security Breaches on the Rise. Revealed August, 2006, from: http://www.snwonline.com/evaluate/database_security_03-31-03.asp?article_id=224.

[8] Tajima, Keishi. “Static Detection of Security Flaws in Object-Oriented Databases.” Proceedings of the 1996 ACM SIGMOD international conference on Management of data SIGMOD '96, Volume 25 Issue 2.