ADO and SQL Server Security

Security is a growing concern in the Internet/intranet development community. It is a constant trade off between access to services and data, and protection of those services and data. This is really the heart of the security dilemmaif your site is *too* secure it could lose its value. On the other hand, without appropriate security measures, you might find someone corrupting your database for you, or selling your company's secrets to your competitors.

Security issues can be addressed at three different levelsthe features offered by the Web server, those offered by the operating system, and those offered by the data source being accessed. In the Microsoft world, the operating system and the Web server are tightly coupled, as are the security features they offer.

Internet Information Server (IIS) supports advanced security facilities, such as a Secure Sockets Layer (SSL) which provides a security scheme for bulk-encrypting data between the server and its clients, when private communication is required. In addition, IIS completely integrates with the object-level and user-level security services provided by Windows NT. This can be used to protect a specific area of your site, such as DSN definition files, to which you don't want people to have access. In Chapter 9, we'll be looking at Web site security in a lot more detail.

However, as powerful as these facilities are, they will generally need to be supplemented with additional security measures that can be used to protect the data accessed by your site. In this section we will take a brief look at SQL Server security, and see what features and capabilities are available in this product. Then we will apply these features to accessing a SQL Server from an ASP page. Finally, we will take a look at a few simple steps you can take to protect your site and its data.

Security and Data Access

The level of security actually required is dependent on the level of access needed by our clients. If our database is servicing a small group of individuals, with information that requires minimal protection, then a very open security framework could be put into place. Essentially, we just need to allow our users to access the data they need, while protecting them from accidentally destroying or damaging it. However, if our database is servicing a public forum, much more stringent controls must be built. Not only do we need to protect the data from accidental damage, we need to verify that those accessing it have the right to do so, as well as protect it against those that do notor who might intentionally damage it.

SQL Server supports several powerful security features that can be applied to each of these scenarios. SQL Server security options determine a server's login security mode, what auditing is done, and what objects, resources and

data a user has access to. In the next section, we will take a high-level look at some of the security features offered by SQL Server.

Login Security

SQL Server 6.5 offers three login security modes. The mode we choose will significantly effect how the server handles security. Let's take a look at each of the options available, and how these options affect our environment:

- **Standard Security Mode**: This is the default security mode. In standard mode, SQL Server manages its own login validation process for all connections (except client applications that explicitly request integrated security over trusted connections).

- **Windows NT Integrated Security Mode:** This mode uses Windows NT authentication mechanisms for all connections. Only trusted connections are allowed into SQL Server. SQL Server always ignores the login name and SQL Server password submitted in the login request from an Open Database Connectivity (ODBC) client application. Network users, who were assigned user-level privileges to SQL Server, log in using their network username or the default login ID (if their network username is not found in **`syslogins`**).

- **Mixed Security Mode**: This mode allows both trusted and non-trusted connections, and is a combination of integrated and standard modes. For trusted connections, SQL Server examines the requested login name as specified by the client ODBC application. If this login name matches the user's network username, or if the login name is null or spaces, SQL Server first tries the Windows NT integrated login rules as described above. If this fails, SQL Server uses the standard rules. If the requested login name is any other value, the user must supply the correct SQL Server password, and SQL Server handles the login using the standard rules described above. All login requests from non-trusted connections are handled using the standard rules.

Integrated and mixed security modes are best utilized in an intranet environment. In this environment, user IDs and privileges are defined in the Windows NT domain, and subject to domain security. The user ID can be retrieved from the client system directly, and authenticated against the domain master list. Here, the user is never required to provide authentication information.

In the big wide world of the Internet, a domain-based authentication model would not be realistic. Here, security must be tightly controlled and access provided only as is necessary.

User Groups

Creating a Login is just the first level of security associated with SQL Server. A Login ID does not permit the user to access any of the objects in a database. Access to a database and the objects within it are granted to individual users or groups of users.

SQL Server security is based on a detailed hierarchy of **groups**, which include **users**. Both groups and users are defined as having specific access to, and control over, services and data. In SQL Server, permissions for services and data can be controlled at a very granular level. For example, access to an individual object can be controlled, and then the actions that are possible on that object can also be regulated.

A group is simply a means of organizing the users of a database. Permissions are assigned to the group, as opposed to individual users. Users in the group have access to any resources available to the group as a whole. This simplifies the administration of users and objects in an SQL Server environment.

There is a built-in group, public, in every database. Each user automatically belongs to public and can be added to only one other group. A user cannot be remove from the public group. If a group is deleted (or 'dropped'), all users in that group are automatically removed from the group. However, dropping a group does not drop its users. Users who were members of the dropped group are still valid users in the database and members of the public group.

Database and Object Owners

SQL Server is organized around databases. Each one contains **objects**, such as tables, stored procedures, rules etc.. Each object has an **owner**, who has full authority over that object. SQL Server recognizes two types of ownerthe database owner (DBO) and database object owner.

The **database owner** is the creator of a database, and has full privileges over it. However, beyond simply having the ability to manipulate the object itself, the DBO has the option of granting access to the database to other users or groups. In summary, the DBO can:

- Allow users access to the database.
- Grant users permission to create objects and execute commands within the database.
- Set up groups
- Assign users to groups and add guest accounts, which give users limited access to a database.

Just like any other user, the database owner logs into SQL Server by using an assigned login ID and password. In their own database, the user is recognized as DBO; in databases which they haven't created, the user is just known by their database username.

As we said earlier, a database contains objects. The user who creates a database object is the **database object owner** for that object. In order for a user to create an object within a database, the database owner must first grant that user permission to create that particular type of object. Just as the database owner can grant permissions for their database to other users, the object owner can grant permissions for their object.

Database object owners have no special login IDs or passwords. The creator of an object is automatically granted all permissions to it. An object owner must explicitly grant permissions to other users before they can access the object. Even the database owner cannot use an object unless the object owner has granted the appropriate permission.

As you can see, database and database object privileges are assigned at a very detailed level. Let's take a look at what privileges (referred to as **permissions**) can be granted to users and groups.

Security Permissions

SQL Server has two categories of permissions: object and statement. Some statement permissions (for the **SELECT**, **UPDATE**, **INSERT**, **DELETE** and **EXECUTE** statements) are handled as object permissions because these statements always apply to database objects that are in the current database.

Object permissions regulate the use of certain statements on certain database objects. They are granted and revoked by the owner of the object. Object permissions apply to the following statements and objects:

Statement	Object
SELECT	Table, view, columns
UPDATE	Table, view, columns
INSERT	Table, view
DELETE	Table, view
REFERENCE	Table
EXECUTE	Stored procedure

Statement permissions are not object-specific. They can be granted only by the system administrator (often referred to as the **sa**) or the database owner. Statement permissions allow the user to create new objects within a database. The following are examples of these statements:

- **CREATE DATABASE** (can be granted only by **sa**, and only to users in the master database)
- **CREATE DEFAULT**
- **CREATE PROCEDURE**
- **CREATE RULE**
- **CREATE TABLE**

- **CREATE VIEW**

Each database has its own independent permissions system. In other words, being granted permission to perform a given task in one database has no effect in other databases.

Now that we have had a chance to take a brief look at SQL Server Security, let's look at a few tips that will help us design our next database.

Security Tips

When designing a database in the SQL Server environment, which will be accessed by public sources, here are a few guidelines to keep in mind:

- Take advantage of the Guest default login. The Guest login is a special login defined by default in SQL Server. Guest has no rights to any object in the server aside from the ability to login. The Guest login can be granted **read** access to the specific views, tables, or objects needed for your Web site. The Web site can then use this account when attaching to SQL Server objects, minimizing your site's exposure.

- Use stored procedures where possible for updates and insertions. Coding static SQL in your ASP page is perhaps the most straightforward way to provide dynamic data content. However, keep in mind that not only can your Web site see this code, but so might a really clever hacker. To minimize this exposure it is often better to call a stored procedure, which in turn performs the required updates or inserts. A stored procedure can be granted permissions to access and modify objects and data that the ID calling the stored procedure cannot. This provides a shield between your site and the actual data being changed.

- When tables are scanned for data, views can serve as security mechanisms. Through a view, users can query only the data provided by the view. The rest of the database is neither visible nor accessible. Permission to access the subset of data in a view must be granted or revoked, regardless of the set of permissions in force on the views underlying tables. Data in an underlying table that is not included in the view is hidden from users who are authorized to access the view but not the underlying table.

Don't expose more then you have to. In most cases, you will want to retain the majority of your data behind the 'firewall', and supply a mechanism to access this data. Always keep in mind that the any security can be compromised and, as such, preventative measures must be in place that assume that this will occur. One recommendation is to store only high-volume transactional data on the SQL Server that is directly accessed from a

Web site. Supportive information can be maintained on a separate secured system, and retrieved as needed using Remote Stored Procedures, or other similar technologies.