The Myth that is Database Middleware Security

Introduction

Synopsis: If we consider the amount of database-driven internet Websites or small to medium sized organizations that employ in-house database technology, itis little wonder that IT managers are often faced with the arduous task of how best to secure their data layer against unwarranted attacks from both internal and external sources.

In this article focusing on data-access middleware, we'll look at a common misconception that surrounds securing a database. This misconception focuses on vunerability in the security aspects of data access middleware.

We only need look at the Slammer virus for instance to understand why the purveyors of this logic have the ammunition they need. But with more businesses being driven to the concept of E-Business, and its many guises, there's no way of denyingthat data-access middleware is becoming an important necessity for organizations today.

Initially, this article focused on E-Business and data access, which in many cases concerns the Internet. But in reality, the internet is only a small part of accessing data. It is true that more organizations are now using content stored within their own internal systems to provide data to users from outside their network but when you think about it, this information existed long before it made its way to the Internet. Bearing this in mind, we can surmise that the Internet is only part of the equation when trying to secure you data. To remove any ambiguity, we'll summarise on what Data Access Middleware really means.

There are several ways of looking at Data Access Middleware. It can be described as a standard database transport interface between an End user application making a database request, processing the data and returning that data back to the end user. In other words the bridge between End user and Database. All this is done using SQL and all communication is carried out over the network

There are several considerations for what happens with data in an organization. Take the following scenario. You might have a secure network but within that network, there are people who could, with a little ingenuity, have access to sensitive data within your database systems. Of course, knowing about this data wouldrequire the user to be privy a few things such as the database being used, the name of the tables that contains this data, and most importantly, have access to the database itself. At OpenLink Software, we cannot legislate for the first two criteria but we can certainly do something about the latter option, Database Access.

Fig. 1

In Figure 1, we can see the different systems that might require access to an organizations database. Some organizations might employ additional sources. With so many entry points, we can begin to see part of the problems facing IT Managers, System Administrators and CTOs when it comes to securing the organizations data layer.

The common denominator between all these sources is that they use some form of Data-access middleware as the means oto accessing the database. Data-Access middleware comprises of (ODBC/JDBC/OLE-DB providers/.NET providers). It is used mainly for its simplicity and accessibility. The inherent problem of data middleware however, is not having its own inbuilt security mechanism.

Some technologies such as JAVA applets, .NET etc, implement their own levels of security that can restrict access at the initial entry point, typically a login screen for example. But not all applications do this.

Authentication

By breaking down the different layers an IT manager will consider when planning their infrastructure, we can begin to see the complexities involved in the decision making process.

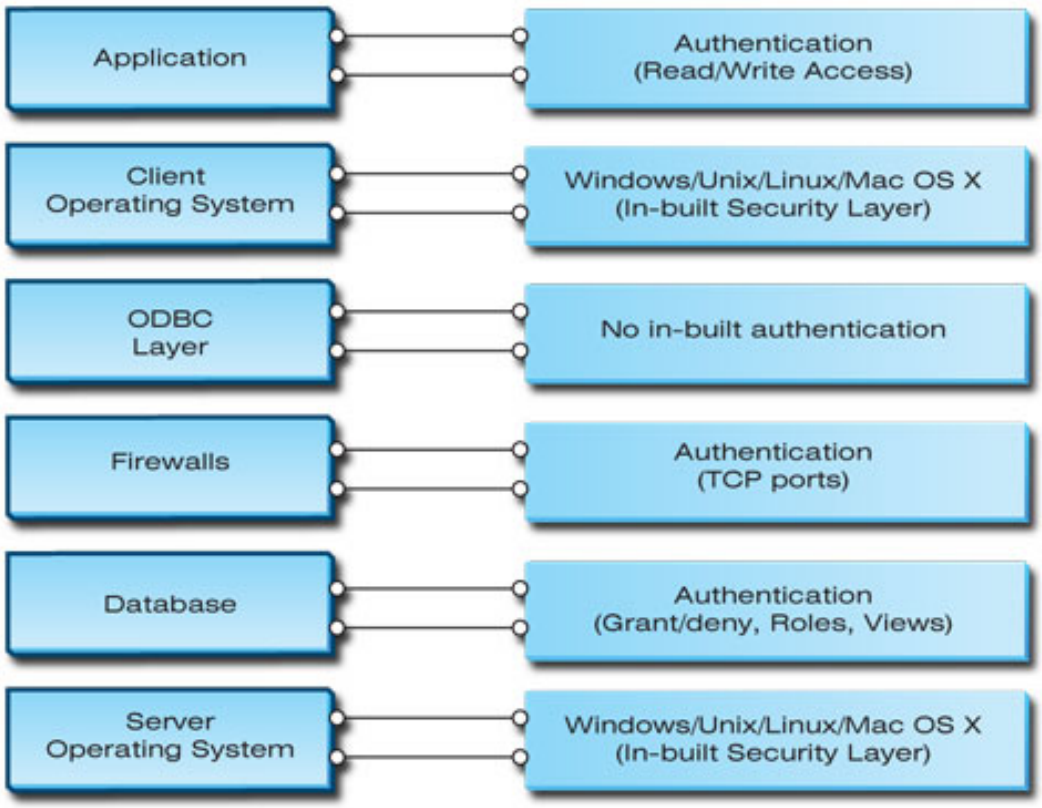

Fig.2

Applications - Tend to be of different types, from Enterprise applications, J2EE & .NET, Application servers, Business Intelligence tools, Bespoke applications. Authentication at this level range from access levels stipulated by typically User Login screens and also by Read Only/Read Write access.

Client Operating System - This is the actual machine that hosts the application being used to access the database. Typically, these are personal workstations. Security at this level revolves around the way client machine is being accessed. Active Directory, Domain users, UNIX authentication, Shadow password access are all different ways the client can interact securely and control access at this layer. The concernhere is, once access has been granted to a user, what the user then does from thispoint onwards could ultimately cause problems.

Data-Access Layer - This layer as indicated earlier does not have its own data security implementation and as such has to rely on the Data-Access driver to provide some form of security implementation. By and large, most do. Every ODBC driver authenticates against the database using the standard methods of database authentication which are typically username/password.

Firewall Access - Firewalls are typically used when restricting access to external users of a network. It achieves this by limiting the entry points on the network where incoming connections are made.

Database Server - The database will have its own form of authentication. Typically, this is the form of Grant/Revoke permissions. Additional authentication can be done by using database views. Another common approach is the concept of Roles (users/groups).

Server Operating System - Databases are usually hosted on servers due to their processing power and memory capabilities. The security at this layer normally uses the same layer as the clients except this controls access or initiates the security to a higher level than the client would.

As Data-Access is middleware, this means there are several factors that can influence its behavior. What is important is how many of the layers described above are made secure. It will never be possible to fully control security at all the layers outlined above but when considering data-access, it is important that consideration is taken for as many of these as is possible, hence why choosing the correct data-access middleware is important.

The OpenLink Software Data-Access drivers are a comprehensive bundle of database connectivity drivers that conform to industry standards such as; Java Database Connectivity (JavaSoft), Open Database Connectivity (Microsoft), and X/Open SQL Call Level Interface (CLI) specification and Wireless Protocol technology.

Through our Client/Server (MT – Multi-Tier) technology, controlling access to your database regardless of size has always been a primary objective in the way it was built. Using a combination of data-access controls, the MT drivers go some way in creating the most comprehensive environment in which data-access is possible. There are several approaches used by this driver.

Please note that these methods apply to the Client/Server software and not our client-only (lite/ST) installation.

Read-Only Authentication:

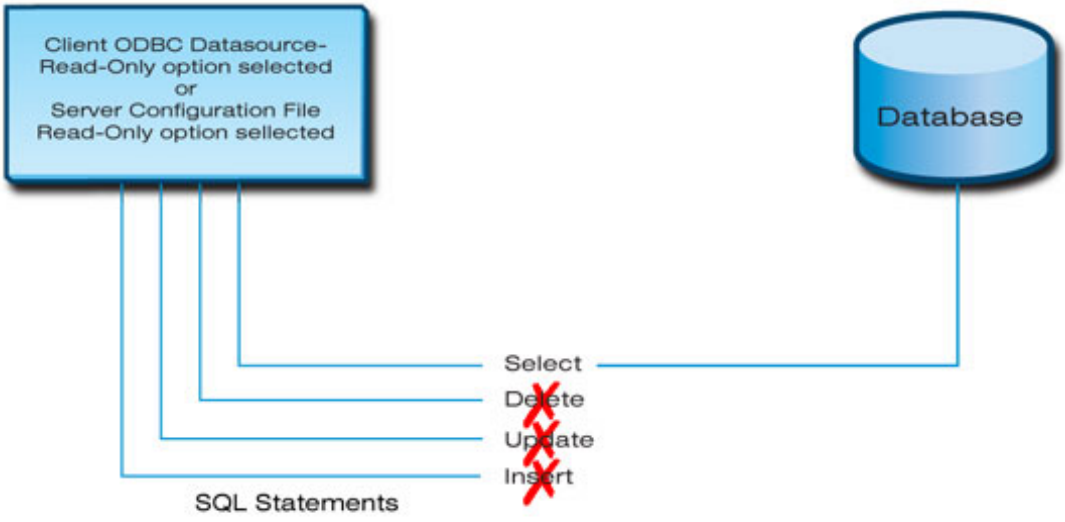

Fig.3

Data security does not always mean restricting who has access to a database. It also concerns the way a user can access information stored within the database. This typically revolves around Insert/Update/Deletion of records. If someone is diligent enough, there

are several ways of bypassing standard protocols and ultimately gaining access to sensitive information. Once this layer has been compromised, the concern will be whether they can edit this information. Restricting a user's ability to update data is a very important consideration. The OpenLink ODBC driver restricts access at both the Client and Server side level.

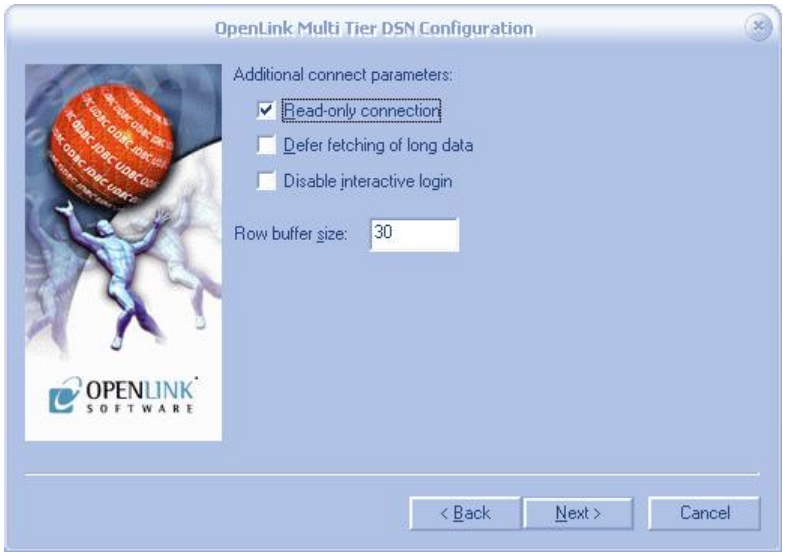

Fig.4

In figure 4, Read-Only access is set-up on the client but any proficient user who has access to this datasource can easily uncheck this option. Therefore, to safeguard against this, there is a server override option. This is activated in the server-side OpenLink configuration file (rulebook).

```
[generic_pro91sql]
Description   = Default settings for Progress 9.1x (SQL-92) agent
ReadOnly   = Yes
```

By setting this flag, regardless of whether or not the client changes their datasource option, the connection will always be read-only.

OpsysLogin

A common method used by Databases to authenticate users is by checking whether the user is a valid user on the Operating System. For instance, SQLServer can validate database users against their Operating system. This means, if the username/password passed by the client application does not validate with what the Operating system expects, the connection will fail.

The MT suite also has this capability and is referred to as OpsysLogin. With this option turned on, any user making a connection will first be validated against the Operating system. If the username and password do not match the OS username and password, access will be restricted. Another advantage of this system is that, it also allows the system administrator some flexibility on how secure you need to make your database. This is because any user who gets past the OpenLink Request Broker authentication layer is required to already have been a valid user.

```
[generic_pro91sql]
Description    = Default settings for Progress 9.1x (SQL-92) agent
ReadOnly    = Yes
OpsysLogin = Yes
```

The OpenLink Request Broker is the main component of the OpenLink Server and handles all incoming and outgoing requests to and from the database. In effect the real Middleware. OpsysLogin is part of the Request Broker's in-built security. Another advantage of having this feature is the ability to interact with C2 Security Servers, e.g. HP-UX. The broker is pre-built with `oplauth', this makes it possible to extend your security layer to use C2 security, Yellow pages when authorizing and authenticating users.

Rulebook Access Control

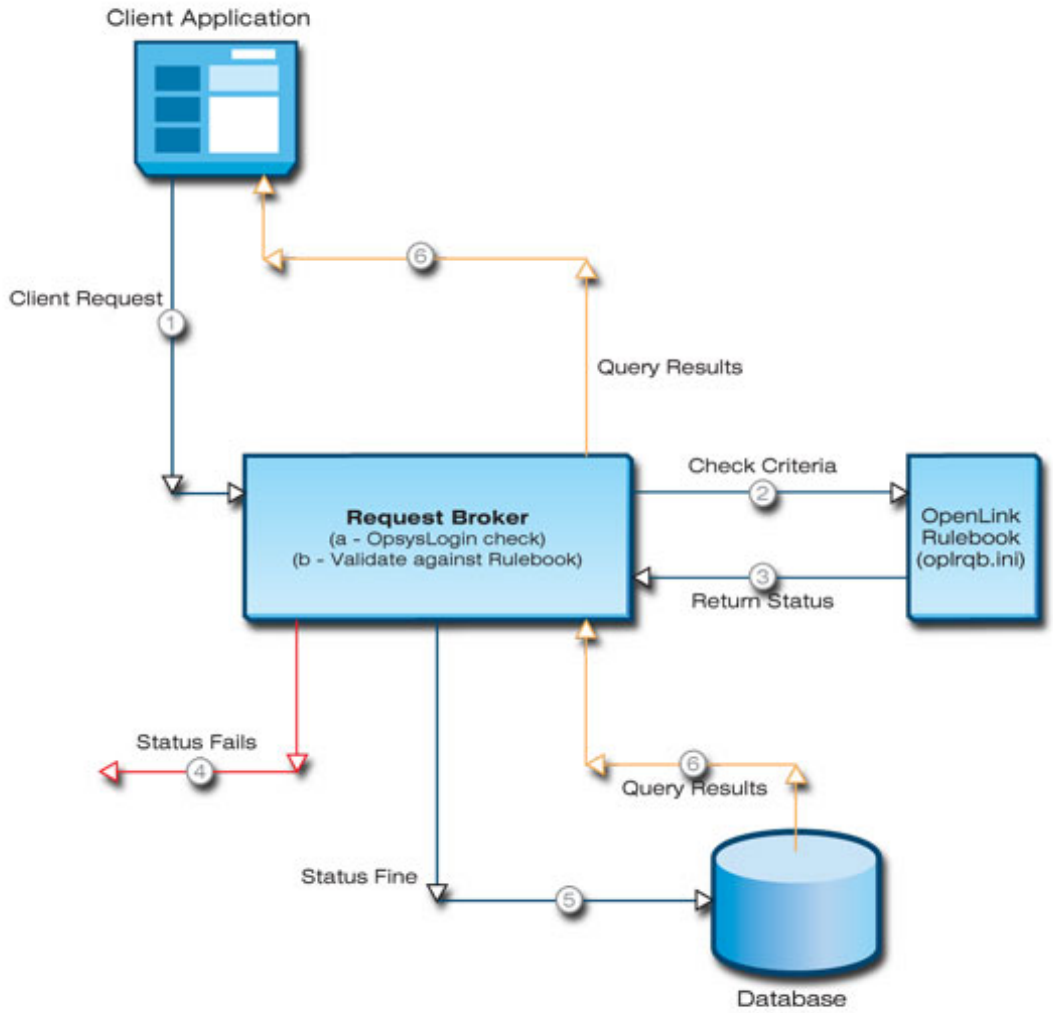

Fig. 5

The OpenLink Server (broker) uses a unique configuration file, commonly known as the rulebook. This is a text based configuration file that controls all aspects of OpenLink's Server component, from security, database configuration to log files etc. It can be edited using either a text editor or via the Web-based administration tool that ships with it.

The Request Broker restricts access based on Rules. As every client connection to the database can only be made through this layer, it is clear to see why control is easy. This is depicted in figure 5.

When the client attempts a connection, it passes, in its datasource information that the broker will use to verify authenticity. When the broker receives this information, it validates this against preset conditions within the rulebook. If the settings match, it then passes the connection onto the next stage in the validation process. If not, the client is notified via a customizable error and a record is made in the log file to inform the administrator.

Below are lists of this criterion. With the exception of the Domain, all the other criteria are optional.

DOMAIN	This is the main identifier and is used to group different connections. Each client connection will have its own a domain as it determines which section of the rulebook the broker must use. This is easily customizable so generic domains can be created, e.g. Sales, Training, Demo can all be names of domains
DATABASE	If this is selected, it means the user must provide a valid database name that matches with a pre-defined list of database names
USERNAME	This verifies the incoming username with a valid user as specified in the Rules but the user does not have to be a valid database user. This means if the user is not in a list of approved users. Useful if the database doesn't have a security model.
OS	This makes it possible to restrict which type of Operating system incoming connections will be accepted. If for instance, your network only uses Windows applications, you can restrict it so that Unix, Apple Mac or Linux client connections are restricted.
MACHINE	It is possible to specify individual machines that are allowed to connect. By specifying their Machine names or IP-Address. Useful if an Application server is been used.
APPLICATION	The same applies to adding specific applications that are allowed to connect. Again, a typical example would be in an Application Server environment where only one application is expected to connect
MODE	Setting this up can restrict access based on Read-Only/Write access as specified in the datasource.

Another unique feature of this option is its ability to group any of these criteria into sections. This means that it ispossible to use aliases within the Rules process. This makes it even easier to update files if conditions change. This method is highly recommended.

```
User Aliases
scott|tiger   =    insecure
```

These rules can be made to be more complex depending on the level of security required by the system administrators. They are case sensitive and follow the Unix Regular expression syntax when setting up aliases.

It was previously explained that the rulebook was a text file. To counteract this, we provide a file called **Security**. This file is a UNIX based file and allows Root permission to be set on the important files located within the OpenLink folder. In other words, if you don't have root access, you cannot edit the file or change the settings. Further preventative measures by your administrator can be used to restrict access to the directory storing the file.

File Permissions before running the Security program as root

```
-rwxr-xr-x  1 openlink users    254736 2003-09-01 12:51  oplrqb
```

File Permissions after running the Security program as root

```
-r-sr-s--x   1  root  root   254736 2003-09-01 12:51  oplrqb
```

Server-side Override:

This is a simple but very effective tool. It allows sensitive information to be stored on the server rather than specifying it on the client. This way, the client machine will never know what parameters should be sent to the server to access the database. This immediately cuts out the prospect of a network sniffer being privy to any sensitive information such as Usernames/Passwords sent over the network. The types of information that can be stored on the server include the following.

```
[generic_pro91sql]
Description   = Default settings for Progress 9.1x (SQL-92) agent
ReadOnly   = Yes
OpsysLogin = Yes
Database = /dbs/Progress91d/wrk91d/sports2000.db
Username = sysprogress
Password = sysprogress
```

The downside to this approach is that the information on the server is been stored in a text format. Consequently, it is down to the system administrator to restrict access to this file. UNIX or Windows permissions apply here.

Proxy Agent:

An effective and common approach employed by organizations is through the use of Firewall technology. If this method is used, there are certain factors that need to be considered when communicating with databases behind the firewall. To assist with this, we provide a Proxy agent. This agent can be used to restrict incoming access via specific TCP ports on the network as has been determined by the Firewall System administrator. This provides a secure communication channel from both internal and external networks.

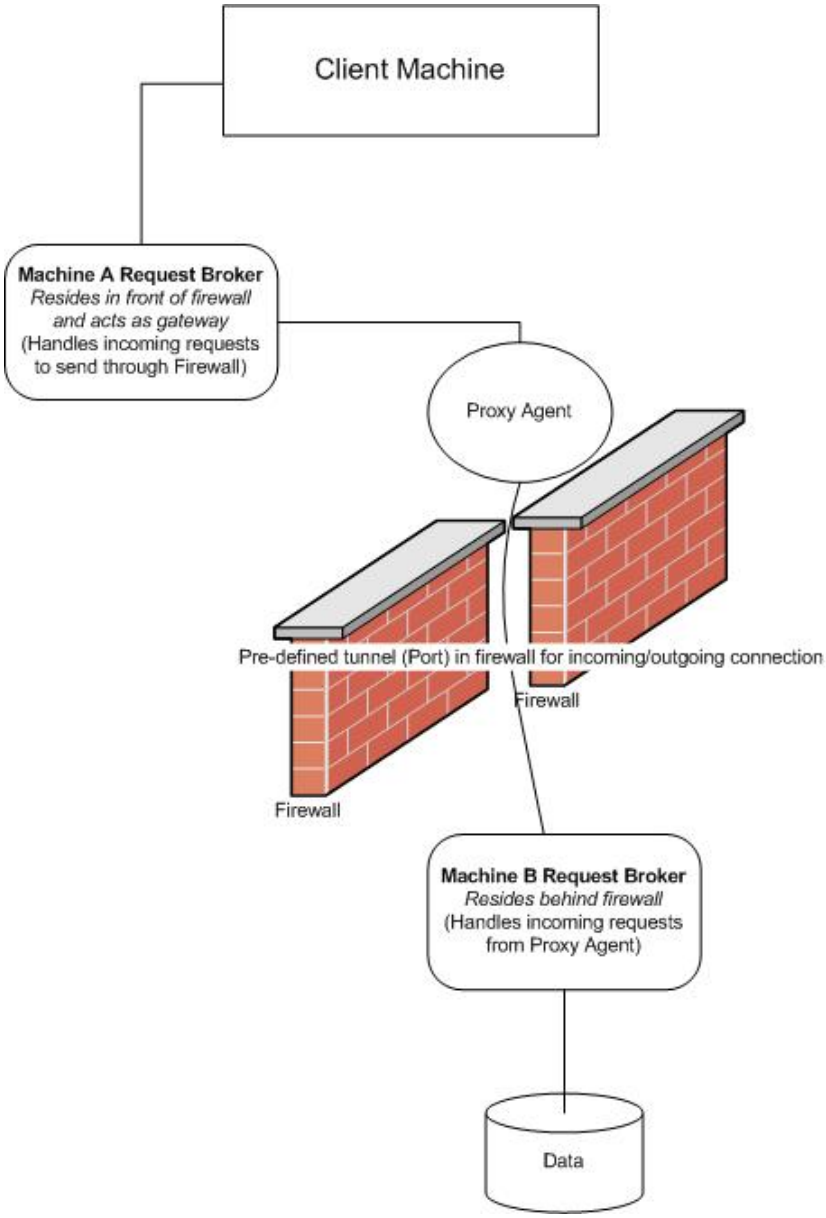

Fig.6

Data Encryption

Using OpenLink's encryption layer, it is possible to encrypt the username/password as well as the data been sent across the transport layer. There are plans to incorporate Data Encryption over SSL.

Conclusion

Although some of the features outlined above are by no means unique to the OpenLink driver, it is without doubt, that not every ODBC driver will provide you with such a comprehensive list as has been illustrated above, and more importantly, in one place.

If all the steps outlined above, allied with application and database security implementation, are incorporated within an IT environment, any IT manager's decision making process on how to secure data will be significantly reduced.