

# **NISCC Technical Note 01/03: Understanding Database Security**

## **Introduction**

This NISCC technical note is intended as an introduction to database security issues. It is vendor-independent and does not promote the use of database technologies. However, some of the URLs referred to in this paper contain advice from particular vendors or from third-party companies.

The intention of the paper is to increase technical awareness about database security in the constituency of the UK Government Computer Emergency Response Team, UNIRAS. Moreover, the scope of paper is limited to the following:

- Assisting potential purchasers to choose a secure database management system
- Assisting database administrators of the database management system to configure the database in a secure way

The note is therefore aimed at:

- Managers, system administrators and security officers in the organisation who wish to inform their decisions on the choice of database management systems that are available
- Database administrators

To assist potential purchasers and administrators checklists are provided at the end of this note.

In this note the term "database" will be used to denote a repository or store of data that has an internal organisation into objects (such as tables, views and procedures). The terms "database application" and "database management system" will be taken to be synonyms, denoting a software product which manages the creation, modification and deletion of database objects (such as databases and users).

This note is not intended to address the issues relating to secure access of a database management system by an application, which will be the subject of a separate NISCC technical note. Instead, it is assumed that input validation has been performed by the application to ensure that the application passes only those parameter values that are appropriate for that application.

In this note it is not assumed that the operating environment of the database management system is benign. For this reason, the note will consider aspects of security external to the database management system itself, although this note should not be taken as a primary reference on those subjects. Readers in organisations in the UK Critical National Infrastructure or in UK Government interested in sources on operating system or network security should consult the NISCC outreach team or seek advice from CESG or a CLAS (CESG Listed Adviser Scheme) consultant respectively. An overview of network security is provided in NISCC technical note 01/02. The following non-exhaustive list of URLs

gives sources of information provided by vendors and reputable, third-party organisations:

- <http://nsa2.www.conxion.com/>
- <http://csrc.nist.gov/publications/nistpubs/>
- <http://www.microsoft.com/security>
- <http://www.sun.com/software/security/>
- <http://www.cisecurity.org/>
- <http://rr.sans.org/index.php>

Database security is an area that has been the subject of a great deal of public attention during the past year. Database management systems are very frequently configured to be available across a network, often using a web page to present the results of a query against the database. The use of database management systems in e-commerce and web application solutions has led to an increased risk of indirect attack on database management systems from the Internet.

Being a computer application, the risks to a database management system are:

- Unauthorised access to confidential data
- Unauthorised modification of data
- Loss of availability of the data
- Ability to compromise the operating system of the database management system to launch further attacks on other computers (as database management systems usually run under an operating system account)

“Data” here includes control and administration information used by the database. To counter these threats, the following are common measures that can be put in place by database applications:

- Identification and authentication of users
- Access control on a per user basis (where possible)
- Accounting of all actions on the database
- Ability to audit the accounting data
- Object reuse (ie erasing the contents of deleted database objects before reuse by other users)
- Database backup and recovery

It needs to be stressed that security measures taken by database management systems are unlikely by themselves to be sufficient to prevent attacks. There are three main reasons for this. Applications run in an environment controlled by the operating system: an application may therefore be vulnerable to attack by exploiting vulnerabilities in the operating system. Another reason is that database management systems are often designed to function in benign environments, where the internal network is restricted to authorised users and any connected networks are assumed to be under an equivalent security regime. It is unlikely that databases designed for this type of environment will be resistant to attack from hostile third parties. Finally, databases frequently come with a rich set of scripts, stored procedures and supporting software, such as application servers for web connectivity. The richness and variety of this functionality increases the likelihood of security vulnerabilities. For these reasons the following additional types of security measures will also be needed:

- Appropriate operating system security
- Appropriate network security
- Removal of unnecessary scripts and procedures on the database server

This note will discuss each of the security measures identified above and the types of vulnerabilities that they need to counter.

## **Identification and authentication of users**

A database application should provide the ability to identify users and authenticate them. There are typically two ways in which identification and authentication can be provided. It can be provided by the database application itself or it can be provided by the operating system. Both of these ways of authenticating users can provide good security, and the choice depends on the strength of the operating system authentication compared to the strength of the database's own authentication scheme. When using operating system authentication, some database applications use operating system identification (eg Microsoft SQL Server), while other correlate operating system user names with user names internal to the database (eg Oracle). On the other hand, with database authentication, database applications use separate user names from those registered by the operating system.

Database applications are increasingly using cryptographic encryption techniques for identification and authentication. It is not uncommon for database applications to support common authentication protocols such as Kerberos, Remote Authentication Dial-In User Service (RADIUS) and Secure Sockets Layer (SSL). If implemented correctly, these cryptographic measures provide stronger identification and authentication than user names and passwords for a database for which remote access is required. A disadvantage of using separate identification and authentication servers is that they are often proprietary to a particular vendor, which may limit the purchaser's choice of solutions.

The primary advantage of database authentication is that it provides an authentication measure to connect to the database which only requires the authentication data to be protected by the operating system. The primary disadvantage of database authentication is that it may allow users to attempt to access the database as any other user, potentially including a system administrator, unless the database management system supports the ability to restrict access to the database based on user name and computer identity [1]. Given that some database applications have built-in administrator accounts with default passwords, the use of database authentication can lead to misuse by authorised users of the system that have access to the database client application.

The primary advantage of operating system authentication is that the user has a single sign-on process in order to access the database, which may be suitable for authorised users of the internal network. The primary disadvantage of operating system authentication is that the user will have to have an account on the operating system, which may have privileges and access rights that an attacker may be able to misuse.

Whilst database identification and authentication can be configured securely, it is common for this area to be weak. Common faults are:

- Lack of requirement for a minimum password length
- Lack of requirement for password complexity
- Database client applications allowing passwords to be supplied in clear
- Failure to remove unused user accounts
- Failure to change default passwords of built-in user accounts

## **Access control**

It is important to control access to the database so that only authenticated users can access the database. Moreover, in some database applications it is possible to control the access at a level below the file permissions that control the operating system user's access. For example, the database administrator may be able to control whether a database user has access to a table, to rows within a table [2], and whether the user can read data or modify the data. Other privileges that a database administrator can grant are the ability to create new database objects (such as tables), the ability to delete database objects and the ability to execute SQL statements and procedures stored in the database. This type of access control is called discretionary access control.

Some databases provide enhanced access control facilities. Other than the fine-grained access control outlined above, there are standards for mandatory access control and role based access control. Mandatory access control is a centrally administered access control model that enforces access according to sensitivity labels as well as database object permissions and user privileges [3]. Role based access control is also a centrally administered access control model, but one that enforces access based on a user's job functions (or roles) [4].

Access control for database applications can sometimes be subverted by incorrect permissions being set at the operating system level. Unauthorised users may be able to access a database as a file and extract data from it by performing a text based search or may be able to modify or corrupt the database. To avoid this potential problem, the operating system should comply with a standard for fine-grained discretionary access control (eg evaluation against the Common Criteria controlled access protection profile [5]). With this degree of access control available at the operating system level, the database files should be owned by a database-specific user and appropriate access should be granted only to operating systems users who are also database users (if any). Database files should only be readable by users authorised to view the corresponding database records.

Even if the database user does not have access to the operating system, it is possible that the system administrator could give a database user inappropriate privileges. At its most severe, the user could have been placed in the database administrators' group, which would allow the user access to the database system objects. Being a database administrator would enable a user to create, delete or modify database objects, including other users' details (including their passwords) and objects that other users own.

Although granting a database user administrative privilege is unlikely in a well managed system, great care should be taken to ensure that users are assigned least privilege needed for their roles: if they only need to read

data from specified tables then only read (SQL "SELECT") access will be needed on only the tables specified; whereas if the user has to add new details then he or she should only be able to write (SQL "INSERT" or "UPDATE") to the specified tables.

From an administrative point of view, where they are supported by the database management system, the following steps can be considered good practice because they minimise the amount of internal database state information available to users:

- Restricting users' access to views rather than the underlying table
- Providing procedures for users to execute rather than executing SQL statements in the user context in order to define the set of queries that the user has available (which does not reduce the likelihood of input validation attacks, but does hide table structures from user view)
- Permitting user database access only via a particular software package (a concept often called "application roles")[6]

### **Accounting of all actions on the database**

In case there is a security incident (either because of an external compromise or internal misuse), it is important to be able to account for all security relevant actions on the database (including operations that have not been permitted by the database management system). For database management systems, the following are examples of types of events that should be accounted [7]:

- Database access attempts
- Access attempts on database objects (eg the database itself, tables, views, procedures)
- Object modification, creation, deletion or execution
- Subject (user, profile, role) modification, creation or deletion

When recording security events the accounting record should contain at least the following information [8]:

- Subject (user, profile, role) identifier
- Time stamp
- Time zone
- Objects involved in event
- Action attempted
- Outcome of action

The security events log should be exportable in a well-structured text based format [9] to other log analysis applications in order to build up a view of a security incident in conjunction with other logs.

It should not be possible to amend security event logs. It may be sufficient to write events to a table readable only by a security officer, although if an attacker gains database administrator access, the attacker could delete all record of the attack. Actions which may mitigate deletion of security logs are:

- Log accounting files to a separate device via a one-way connection

- Copy security logs on a frequent and regular basis to a central logging server by means of a one-way connection [10]

## **Audit of the accounting data**

One of the largest problems with reviewing accounting logs is the lack of auditing tools available with database management systems that produce comprehensible reports. It is strongly desirable that database management systems should have the capability to analyse the audit data to produce human-readable, readily intelligible reports or at a minimum that the accounting logs can be exported in a structured text based format to be read by a log analysis application. In order for security officers to be able to audit logs regularly and successfully, it should be easy to report various aspects or "slices" of the data and to produce summaries of the data in the logs.

For databases the following reports, for example, are desirable:

- Database authentication success and failures grouped by user name
- Access failures grouped by user name
- Access failures and successes to database system objects grouped by user name
- Access failures and successes grouped by database object
- Procedures executed grouped by subject identifier
- Modification, creation and deletion of database objects, users, roles and profiles grouped by user name
- Database activity within a particular timeframe

## **Object Reuse**

Object reuse security is an aspect of database security that users need not be aware of. Object reuse security is the zeroing or blanking of all objects created by one user and subsequently deleted. It is a feature of the database that only becomes apparent if objects are not erased before reuse. An extreme example of the failure of object reuse security would be that one user deletes a database table and a different user creates a table of the same name and recovers some of the data in the previously deleted table. This could clearly undermine the confidentiality of data held in the database.

Object reuse security is clearly an essential requirement on any database. It is part of a broader class of security functionality sometimes called separation of security domains. What this means is that a particular subject (user, role, profile or process) should only be able to access objects for which the subject is authorised (sometimes called objects in the subject's security domain). In particular, deleted objects belong to the system and should only be recoverable by the database administrator.

## **Database Backup and Recovery**

As with any information asset, it is important for business continuity as well as security to back up mission critical data. Most database applications have backup and recovery suites available from the product vendor, and for the common applications third party backup and recovery products are also available. With some database applications it is also

possible to recover a database from a known good state by replaying the redo logs.

Whichever technique is used, it is important to schedule regular backups at a frequency that suits the business needs of your organisation. It is also extremely important to verify the success of those backups and to exercise the recovery mechanism periodically.

It is also desirable that a database supports transactions, being able to commit an update to the database and to undo or "rollback" database updates.

## **Operating System Security**

A database will only be as secure as the environment in which it sits if it accepts queries from or mediated by that environment. Confidentiality needs should determine the type of security provided the operating system: whether that is fine-grained discretionary access control, mandatory access control or role based access control, as outlined in the section on access control. Mandatory and role based access control can be used to enforce differing security policies for different types of user. In contrast, discretionary access control allows the administrator to set permissions and privileges for each user individually and for each user to assign permissions on objects that the user owns.

Discretionary access control is more prone to inappropriate file permissions, but role based and mandatory access control can be more difficult to configure. It is therefore essential to check the correctness of the permissions and privileges assigned to users and roles and that networking access is also correct for them. There are number of tools provided by operating system vendors and by third parties (including a number of vulnerability scanners) that can be used to assist the system administrator in enforcing the organisational security policy.

Removal of all unnecessary application and network servers (such as FTP and telnet) from the database server host is also strongly recommended in order to prevent unauthorised access to the database server via these services.

## **Network Security**

Details of protecting a network against attack are provided in NISCC Technical Note 01/02. In the case of database management systems in particular, if access is needed to database from external networks, a firewall should be used to filter network access to the database server and to ensure that traffic is only forwarded to the port(s) on which the database server is listening. Using an application proxy firewall that provides a proxy for the application level protocol used to communicate SQL to and from the database [11] may also enhance security. Application level proxies may provide protection against malformed requests and buffer overflows. If external access to the database management system is not needed, the firewall should be configured to block all traffic to the database management system port(s).

A network intrusion detection system could be deployed to detect attacks against a database server, although successful detection will depend on the placement of the sensor. Some advice on types of intrusion detection

system and placement of sensors is provided in NISCC Technical Note 05/02.

Regular checks with a database vulnerability scanner may also detect vulnerabilities; but these tools are most effectively deployed as part of a security audit by suitably qualified consultants (such as those on the UK Government's CHECK scheme).

## **Removal of Unnecessary Scripts and Procedures**

Stored procedures are often provided with databases to provide enhanced functionality for system administrators and application developers. For this reason they should not be available to end users of the system. If a stored procedure is not used at all and dropping it has no effect on the system's operation, then the stored procedure should be dropped. If they are potentially useful to administrators or developers, "EXECUTE" permission should only be granted to those users or roles that need to be able to run those procedures. The reason for this is that stored procedures sometimes run in highly privileged user context (such as database administrator) and may enable users to violate the organisational security policy.

Since they have been subject to a great deal of discussion in the literature [12], it is worth mentioning two examples of stored procedures that can be abused. The first of these is "xp\_cmdshell" provided with Microsoft SQL Server. "xp\_cmdshell" enables the database user to run commands on the operating system of the database server with the permissions of the operating account running SQL Server (providing that the database user has been granted permission to execute the procedure) [13]. This can assist in compromising the host operating system. The other stored procedure is OWA\_UTIL.cellsprint provided in Oracle Web Application (OWA) utilities package that is installed as part of the Oracle PL/SQL Web Toolkit [14]. This procedure allows arbitrary SQL commands to be run and a specified number of rows returned. Clearly this can be abused to extract information from tables in the database, potentially including administrative information (such as user names) [15].

These two procedures give a flavour of the types of functionality that may be available. It is advisable to list procedures that are stored in the database [16] and check whether they are needed. Procedures can then be dropped or their permissions altered appropriately.

## **Recommendations**

As a general recommendation, system security officers should monitor the publications of vulnerabilities in the database management system used and should apply the vendor supplied patches and workarounds as soon as they have been proven to be stable. (Installation on a reference or test system will help determine stability of a patch in a similar environment to the operational one.) This same advice applies to operating system and to any network infrastructure used by the database.

The other aspects to maintaining database security are those discussed in the previous sections. These aspects should be addressed in your organisation's security policy. Of all those aspects, access control is probably the most significant in addressing the threat to data confidentiality.

Tools such as database and network vulnerability scanners and intrusion detection systems can be very useful in assessing and reducing the risk of attack to databases. These tools can be used by system administrators, although for maximum effectiveness they should be used as part of a security audit performed by suitably qualified consultants (such those accredited under the UK Government's CHECK scheme).

Two checklists of the issues raised in this note are provided below. The issues relate to database security functionality and configuration and do not relate to any assurance measures that the functionality has been implemented correctly.

### **Purchaser Checklist**

Does the database comply with an independent, third party security standard?
Does the database vendor issue security advisories to customers when vulnerabilities are discovered with suggested solutions?
Does the database vendors publish patches in a timely fashion to fix known vulnerabilities?
Does the database management system support operating system authentication?
Does the database management system support database authentication?
Does the database management system enforce strong password authentication schemes?
Does the database management system support cryptographic authentication methods such as Kerberos or SSL?
Does the database management system support application roles to allow users to access the database only via specified software applications?
Does the database management system allow SELECT (read), INSERT/UPDATE (write) and execute permissions to be assigned to database objects?
Does the database management system allow access to be controlled by row label?
Does the database management system support mandatory access control?
Does the database management system support role based access control?
Does the database management system record the subject and object involved in a database event, the action attempted, the result and the timestamp?
Does the database log all actions performed on the database?
Does the database enable security reports on the accounting logs to be produced in an easily comprehensible format?
Does the database enforce data separation and object reuse?
Does the database provide backup and recovery functionality?
Does the database provide rollback of transactions?
Does the database operate on an operating system which supports fine-grained discretionary access control or mandatory or role based access control?
Is the database data transfer protocol supported by an application proxy firewall?
Can stored procedures be identified and those not needed dropped or assigned permissions so that only appropriate users can execute them?
Does the vendor supply secure configuration guidance for the database management system?

### **Database Administrator Checklist**

Enforce requirement for a minimum password length
Enforce requirement for password complexity

Consider using cryptographic authentication techniques
Remove unnecessary user accounts
Change default passwords of built-in accounts
Assign users the appropriate privileges for their roles
Revoke user access to database objects that they do not need
Restrict user access to database objects according to their role
Restrict users' access to views rather than the underlying table
Where possible, provide procedures for users to execute rather than executing SQL statements in the user context in order to define the set of queries that the user has available
Where possible, permit user database access only via a particular software package
Log all actions relating to the database
Analyse the accounting logs (with a third-party tool if necessary)
Plan and implement a backup and recovery regime
Harden the operating system by removing unnecessary applications and network servers
Harden the operating system by setting appropriate permissions on all database files
Drop all stored procedures that are not needed
Restrict user access to stored procedures according to their role
Monitor patches for vulnerabilities and install patches as as they have been proven to be stable
Use vulnerability scanners and configuration checking software from vendors or reputable third parties to provide initial assessments of the security of the database management system

## Notes

[1] A number of database management systems provide the ability to restrict access to a database server by client IP address, but some open source database management systems provide the facility to restrict access by database user name and IP address.

[2] Access control based on a row security label can be used to implement mandatory and role based access control.

[3] Mandatory access control is a term derived from the B class (B1 and B2) in the US Trusted Computing Security Evaluation Criteria (TCSEC). The Common Criteria Labeled Security Protection Profile explains the requirements of mandatory access control to operating systems, see [niap.nist.gov/cc-scheme/PP\\_LSPP\\_V1.b.html](http://niap.nist.gov/cc-scheme/PP_LSPP_V1.b.html).

[4] See [http://www.cesg.gov.uk/assurance/iacs/itsec/documents/protection-profiles/media/RBAC\\_987.pdf](http://www.cesg.gov.uk/assurance/iacs/itsec/documents/protection-profiles/media/RBAC_987.pdf) for a certified Common Criteria protection profile for role based access control.

[5] Discretionary access control is a term derived from the C class (C1 and C2) in the US Trusted Computing Security Evaluation Criteria (TCSEC). The Common Criteria Controlled Access Security Protection Profile (CAPP) explains the requirements of mandatory access control to operating systems, see [niap.nist.gov/cc-scheme/PP\\_CAPP\\_V1.d.html](http://niap.nist.gov/cc-scheme/PP_CAPP_V1.d.html).

[6] Application roles typically impose another level of authentication. They are not a fundamentally new security mechanism, but they can be a useful defence in depth measure.

[7] See the auditing requirements in CAPP (see previous note) and the Database Management Protection Profile (DMPP) (<http://www.cesg.gov.uk/assurance/iacs/itsec/documents/protection-profiles/media/dbms.pp.pdf>) FAU\_GEN.1.1 Security Functional Requirement for example.

[8] Compare the auditing requirements in CAPP and DMPP (<http://www.cesg.gov.uk/assurance/iacs/itsec/documents/protection-profiles/media/dbms.pp.pdf>) in FAU\_GEN.1.2 and FAU\_GEN.2.1 Security Functional Requirements for example.

[9] The use of XML would be a structured format as would a comma separated text file with field identifiers.

[10] Taking copies of audit logs is unlikely to capture a short-duration automated attack, but it does provide a means of detecting non-scripted attacks and persistent attacks.

[11] The Borderware Firewall Server Version 6.5 and Gauntlet Internet Firewall for Windows NT Version 3.01 are products certified under the UK ITSEC Scheme which included SQL proxies for Microsoft and Oracle and Microsoft database communication protocols respectively, see <http://www.cesg.gov.uk/assurance/iacs/itsec/cpl/prodres.cfm>.

[12] See, for example, SPI Dynamics white paper "SQL Injection: Are Your Web Application Vulnerable?" available from <http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf>, and Chris Anley's paper "Advanced SQL Injection In SQL Server Applications" available from [http://www.nextgenss.com/papers/advanced\\_sql\\_injection.pdf](http://www.nextgenss.com/papers/advanced_sql_injection.pdf).

[13] See [http://www.nextgenss.com/papers/advanced\\_sql\\_injection.pdf](http://www.nextgenss.com/papers/advanced_sql_injection.pdf) for details of xp\_cmdshell and other extended procedures.

[14] See David Litchfield's "Hackproofing Oracle Application Server (A Guide to Securing Oracle 9)", available from <http://www.nextgenss.com/papers/hpoas.pdf>. Other Oracle packages that need careful configuration include many of the utility packages such as UTL\_FILE (operating system file access) and UTL\_TCP (TCP network connections). See [http://otn.oracle.com/deploy/security/pdf/ias\\_modplsql\\_alert.pdf](http://otn.oracle.com/deploy/security/pdf/ias_modplsql_alert.pdf) for more information.

[15] Tests on an Oracle 8i database indicate that OWA\_UTIL procedures when run from a SQL client (rather than via an application server) are not able to access objects belonging to other users unless they have been explicitly granted permissions on those objects. Thus access to administrative information will not be available by default but it may have been misconfigured.

[16] Checking the installed stored procedures will vary from database management system to database management system. For Microsoft SQL Server see <http://support.microsoft.com/default.aspx?scid=KB;EN->

[US;q154756&FR=0](#). For Oracle, "SELECT object\_name FROM all\_objects WHERE object\_type = 'PROCEDURE';" and "SELECT package\_name, object\_name FROM user\_arguments;" will identify packaged procedures and unpackaged public procedures.