ICT Standards

Database Security Guide

Document number: ISMF-ICT/3.03 - ICT Security/MISP/SD/DBSec
Version: 1.10
1 Document control
1.1 List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISP</td>
<td>Ministry Information Security Policy</td>
</tr>
</tbody>
</table>
1.2 Purpose of this Document

The Database Security Guide is an attachment to the MISP Policy and MISP Handbook. It is part of the ICT Security package that has been produced within the scope of the ICT Standards project. This project is one of the three sub-projects executed under the global project name “Software Development and Technical Assistance for NISFED, e-Government and ICT Standards Applications”, started 20/08/2006 and rolled out within the scope of the ISMF programme.\(^1\)

\(^1\) For a complete list of documents related to the ICT Standards project, refer to the Project Master Plan; ISMF-ICT/3.01, V.2.00.
2 Introduction

One of the more recent evolutions in network security has been to move away from protecting the perimeter of the network to protecting data at the source. The reason behind this change has been that perimeter security no longer works in today's environment. Today, more than just employees need access to data. Essentially, partners and customers may need access to the data as well meaning that a database cannot simply be hidden behind a firewall. Of course, as databases become more exposed to Internet, it is imperative that these are properly secured from attacks from the outside world. Securing databases involves not only establishing a strong policy, but also establishing adequate access controls.

2.1 Background

Database applications are among the most common systems in the Ministry environment and may pose risk to the confidentiality and integrity of sensitive information. It is therefore critical that developers and administrators manage database systems securely. This document outlines a series of standards and an approach to database security.

This document presents two security postures:
- One applies to any database system
- The other raises the security bar based on application risk. In most cases, application risk is a function of the sensitivity of information stored in a database.

This document distinguishes sensitive or "Restricted" information, from less sensitive or "Unrestricted" information. While it may be appropriate to require higher levels of security for systems that store or process unrestricted information, for simplicity purpose we will use the same terms in this document. Hence, a database or database system that stores or processes restricted information (e.g. a company's performance report) is considered here a "Restricted System." In several places in this document, stronger security standards are explicitly stated for Restricted Systems. It may be appropriate to implement similar security controls on systems that fall outside the "Restricted" area, and it is therefore recommended that each standard is evaluated by developers or administrators of any system.

There is no way that a set of standards designed for wide distribution across a diversity of systems can capture every consideration necessary to secure a system. System security relies upon a number of complex operational, physical and technical variables, many of which may be unique to a given system. There is no substitute then for a careful and continuous assessment of systems risk and documentation of security measures. Equally important is the knowledge and experience of the host and database administrators. Managing a secure SQL Server deployment requires a solid grounding in the technology, and thus a certain knowledge base is assumed for anyone reading this standard or, in turn, intending to manage a deployment in the Ministry environment.

2.2 Audience

The target audience for this document is the Ministry staff responsible for developing, administering or evaluating database-driven applications.
3 Understanding Vulnerabilities

In order to understand vulnerabilities, we should start by describing the various classes of vulnerabilities:
- Vendor bugs
- Poor architecture
- Misconfigurations
- Incorrect usage

3.1 Vendor Bugs

Vendor bugs are buffer overflows and other programming errors that result in malformed commands doing things they should not have been allowed to do. Downloading and applying patches usually fix vendor bugs. To ensure you are not vulnerable to one of these problems, you must stay aware of the patches, and install them immediately when they are released.

3.2 Poor Architecture

Poor architecture is the result of not properly factoring security into the design of how an application works. These vulnerabilities are typically the hardest to fix because they require a major rework by the vendor. An example of poor architecture would be when a vendor utilizes a weak form of encryption.

3.3 Misconfigurations

Misconfigurations are caused by not properly locking down databases. Many of the configuration options of databases can be set in a way that compromises security. Some of these parameters are set insecurely by default. Most are not a problem unless you unsuspectingly change the configuration. An example of this in Oracle is the REMOTE_OS_AUTHENT parameter. By setting REMOTE_OS_AUTHENT to true, you are allowing unauthenticated users to connect to your database.

3.4 Incorrect Usage

Incorrect usage refers to building applications utilizing developer tools in ways that can be used to break into a system. SQL INJECTION is an example of incorrect usage.
Example Issues with Database Security

4.1 MS Access Database
Access, the security, scalability and reliability of such database platforms are dubious. We therefore strongly recommend that all such systems, especially those operating in a production environment requiring high availability, move to an enterprise database platform such as Oracle or Microsoft SQL Server.

4.2 Oracle Security

4.2.1 Listener Service
A good place to start delving into Oracle security is the Listener service - a single component in the Oracle subsystem. The listener service is a proxy that sets up the connection between the client and the database. The client directs a connection to the listener, which in turn hands the connection off to the database. One of the security concerns of the listener is that it uses a separate authentication system. It is controlled and administered outside of the database, and runs in a separate process under the context of a privileged account such as ‘oracle’. The listener accepts commands and performs other tasks besides handing connections to the database.

4.2.2 Listener Security Is Not Database Security
Why is the separation of listener and database security a potential problem? There are a few reasons. First, most people do not even realize that a password must be set on the listener service. The listener service can be remotely administered just as it can be administered locally. This is not a feature that is clearly documented, and is not well known by most database administrators.
Secondly, setting the password on the listener service is not straightforward. Several of the Oracle8i versions of the listener controller contain a bug that causes it to crash when attempting to set a password. You can manually set the password in the “listener.ora” configuration file, but most people do not know how, nor do they have any idea that they should. The password itself is either stored in clear text or as a password hash in the listener.ora file. If it's hashed, setting the password in the listener.ora file manually cannot be performed. If it is in clear text, anyone with access to read the $ORACLE_HOME/network/admin directory will be able to read the password.

4.3 Microsoft SQL Server Security

4.3.1 Collecting Passwords
When SQL Server is running using mixed-mode authentication, login passwords are saved in various locations. Some passwords are saved using strong encryption and permissions (such as the passwords saved in master.dbo.sysxlogins), but many of them are saved using weak encryption (most accurately referred to as encoding) and weak default permissions. You may be asking, “Why are passwords saved with weak encryption?” The reason is that these passwords must later be extracted and used by SQL Server to establish connections with itself and other SQL Servers. This occurs during many of the batch processes that SQL Server relies on, including replication, jobs scheduled through the SQL Agent, and Data Transformation Services (DTS) packages.
Using various techniques, such as examining system tables and stored procedures or even through running tools such as SQL Profiler, it can be determined where and how these passwords are saved. Typically system tables that hold these passwords are properly secure, that is only if the user has permissions to select from the table. There are, however, system stored procedures that access these tables - so looking at these stored procedures is a good place to start.

**Recommendations**

- Keep SQL Server up to date with security fixes
- Use Integrated Authentication
- Disallow Cross-Database ownership chaining
- Run SQL Server under a low privileged account
- Set SQL Server Agent Alerts on critical issues
- Run periodicals checks on all system and non system objects (tables, views, stored procedures, extended stored procedures) permissions
- Run periodicals checks on users permissions
- Audit as much as you can
5 Monitoring Tool

One fundamental of a secure system including databases is thorough monitoring of things like access, log-in attempts, database changes, change management, OS configurations, etc. Monitoring systems extends well beyond security, and it is often a good idea to leverage, for example, tools or practices used for performance monitoring when thinking about security. Administrators of databases and applications are required by the MISP Policy to monitor according to the inherent risk of the system. This may be as simple as occasionally checking privileges and access control lists or as elaborate as record-by-record logging that include triggers for anomalous events. For system servers, monitoring with Microsoft Operations Manager (MOM) or Tripwire may indicate potential security issues. There are also a number of third party products for monitoring applications, databases and hosts. However, please note that the most important monitoring of any system is the expertise and experience of the administrator, and there is no substitute for routine reviews of logged materials by knowledgeable professionals.
6 Mandatory Security Standards for All Systems

6.1 General

- All multi-user, business critical or restricted databases must be inventoried at the appropriate level. Additionally, some description of database contents is required; detailed descriptions will be required when the database contains financial information.
- Servers and host systems on database and application hosts must be configured and administered according to current standards including Windows 2000 and 2003 Server Administration and Security Standards.
- Physically secure database servers and their backup media in locked, access-controlled rooms, through cable locks, cabinets or other similar controls.
- Disable services or features of the operating system and database server that are not in use.
- Make sure that all data and system files are installed on appropriate partitions and that the appropriate ACLs are applied. If someone should gain access to the OS, make sure that the necessary permissions are assigned.
- Never permit user applications to send arbitrary SQL commands to the server without the application serving as an intermediary, and never permit public Internet-accessible applications (such as web applications running on IIS) to send user-defined SQL commands to the back-end database, even with input validation rules.
- Database should be installed under an administrative account.
- Document all data interfaces, including applications, databases and messaging links.
- All default and null passwords should be changed to passwords compliant with the MISP policy.

6.2 Access Control

- Administrative access must be individual user accounts and not a shared group account.
- Neither user nor administrator credentials may be transmitted in clear text. Consider using SSL for encrypting application traffic between clients and database servers or using IPSec for encrypting communications with the server.
- Assign a long and complex password to administrative accounts.
- In systems where users are directly accessing the database server using either client utilities or an application, directory services should be used. Using Active Directory will automatically enforce password aging and strength rules.
- Delegate authority over the server and its databases by utilizing fixed server and database roles appropriately, or by creating own custom roles, so that excessive power is not being placed in the hands of those who don’t need it. In short, do not simply place everyone in the administrator role out of convenience. Also avoid granting rights to individual accounts rather than groups.
- All database and server management scripts must be stored in an access-controlled folder or directory. Audit all access. Scripts should not contain hard-coded passwords. Service OS accounts should not allow a direct login.
6.3 Monitoring

- Regularly audit all shared folders on the database server to ensure that all permissions are minimal and that none of the shares are unnecessary. Be especially wary of shared folders that permit Write access.
- Regularly audit the membership of the administrator role and the passwords of all accounts and logins in it.
- Document grants of elevated database permissions.
7 Mandatory Security Standards for Restricted Systems

7.1 General
- Database servers should have as little accessibility and visibility to the Internet as possible. When, for example, an Internet-accessible web server is used as a front end for a database application, the database should not be on the Web server host itself. In addition, the database host or network firewall should deny all traffic except for specific, static IP addresses and ports of application and interface servers.
- Database server and application server host register in the “Tenable” vulnerability scanning system and be subject to routine security scans.
- Back-up files should be encrypted where possible (especially when transporting off-site). If the database is encrypted, the back-ups will usually be encrypted also. If not, consider using back-up tools that will encrypt data and utilize practical key management solutions.
- Application interfaces should include banners in accord with MISP policies discussing user responsibilities regarding confidentiality and privacy of restricted data.

7.2 Access Control
- Consider using automated tools, group privileges and policies should be used to enforce common sense security precautions, such as disabling null user session access and renaming built-in administrator accounts.
- Avoid hard coding passwords into connection strings in database applications.
- Consider removing the local administrative groups from the database roles and replacing it with a custom local group with only true database administrators. This may not prevent local administrators from granting themselves any access they wish, but at least these actions would be auditable.

7.3 Encryption
- Database files with restricted information stored on mobile devices (e.g. laptops, back-up tapes) or at risk workstations (e.g. in public areas) must be secured through encryption and strong passwords -- or equivalent authentication.
- Protect credentials (and similarly highly sensitive restricted information) through encryption. Consider using SSL certificates.
- When transmitting data or replicating databases across an un-trusted network, encrypt data (e.g. SSL, point-to-point VPN tunnels).

7.4 Monitoring
- Document a host and application audit plan that should include new threats, vulnerabilities, and object access failure statistics (e.g. MOM, Tripwire).
- Regularly audit user group or role access.
- Regularly audit the execute permissions on stored procedures. User logins rarely need execute permission. When in doubt, grant permission only to the administrator roles.
8 SQL Injection

Just because your database is behind a firewall does not mean that you do not need to worry about it being attacked. There are several other forms of attacks that can be executed through the firewall. The most common of these attacks today is SQL Injection. SQL Injection is not an attack directly on the database. SQL Injection is caused by the way web applications are developed. Unfortunately, since you are trying to protect the database, you need to be aware of these issues, know how to detect them, and fix the problems.

SQL Injection works by attempting to modify the parameters passed to a web application to change the SQL statements that are passed to the database. For instance, you may want the web application to select from the ORDERS table for a specific customer. If the hacker enters a single quote into the field on the web form, and then enters another query into the field, it may be possible to cause the second query to execute.

The simplest way to verify whether or not you are vulnerable is to embed a single quote into each field on each form and verify the results. Some sites will return the error results claiming a syntax error. Some sites will catch the error, and not report anything. Of course, these sites are still vulnerable, but they are much harder to exploit if you do not get the feedback from the error messages.

This attack works against any database. How this attack works varies slightly from database to database, but the fundamental problem is the same for all.

8.1 Preventing SQL Injection

Preventing SQL injection from happening is simple once you understand the problem. There are really two strategies you can use to prevent the attacks:

- Validate User Input
- Use parameterized queries

Validating user input involves parsing a field to restrict the valid characters that are accepted. In most cases, fields should only accept alphanumeric characters.

Also you can escape single quotes into two single quotes although this method is riskier since it is much easier to miss parsing input somewhere.

Using parameterized queries involves binding variables rather than concatenating SQL statements together as strings.

The biggest challenge will be reviewing and updating all the old CGI scripts, ASP pages, etc… in your web applications to remove all instance of this vulnerability. It is also suggested that you setup a programming guideline for web programmers that includes emphasis on using parameterized queries and not constructing SQL by concatenating strings with input values.
9 Conclusion

There are a few simple tasks that can be performed to reduce your security risk at a reasonable level:

- Stay patched
- Stay aware of database security holes
- Ask questions on database security, check out
- Explore possible third-party solutions

Provide multiple levels of security:

- Perform audits and pen tests on your databases regularly
- Encryption of data in motion
- Encryption of data at rest within the database
- Monitor your log files
- Implement intrusion detection

9.1 Recommended Security Practices

- Up-to-date anti-virus protection on all servers. While it is good practice to routinely scan files, it may be prudent to exclude folders which contain database files, transaction logs, snapshot files, or other similar files that are unlikely to be infected (and which would entail significant performance penalties if frequently scanned)
- Stored procedures, functions and views should mediate and regulate interaction with the database to validate requests and to hide the details of the database's design from the user application
- Database servers should be configured so that network firewall can control incoming and outgoing packets. Consider regulating traffic from the internal LAN as well. Database servers use specific ports for data sessions and handshaking
- Disable any unused network libraries. If you have a choice in the design of your client applications, design them around the TCP/IP net library
- Where possible, encrypt the code of stored procedures, triggers and views using the "WITH ENCRYPTION" clause when creating them. This is especially important for turn-key database solutions or when you fear physical compromise of the server or its backup media
- Web access to restricted data should be conducted through VPN connection or through and internal IP address