Database auditing is the activity of monitoring and recording configured database actions from database users and nondatabase users, to ensure the security of the databases.

An administrator can perform auditing on individual actions, such as the type of Structured Query Language (SQL) statement executed, or on combinations of data that can include the user name, application or time stamp, for example. Auditors need to audit for both successful and failed activities, and include or exclude specific users from the audit:

Proper auditing of a database will ensure the safeguarding of the database, which means that the database and its features’ installation, default account, patches, services, password policy, account lockout policy and audit policy have proven auditing to be a continuous process.

The major types of risk activities include:
- **Mistake**: Failure to maintain or operate the database as required leads to accidental disclosure of information, and unauthorized changes lead to unauthorized and accidental disclosures, inserts, updates or deletions.
- **Misuse**: Failure to maintain access rights to the database leads to abuse of privileged access and leakage of information.
- **Malicious action**: Failure to maintain a secure, logical setup of the database leads to data theft or a denial-of-service (DoS) attack.

**COMMON VULNERABILITIES FOUND IN DATABASE ATTACKS**

Many attacks begin with social engineering:
- **Phishing** is an e-mail fraud method in which the perpetrator sends out legitimate-looking emails in an attempt to gather personal and financial information from recipients. Typically, the messages appear to come from well-known and trustworthy web sites. Web sites that are frequently spoofed by phishers include PayPal, eBay, MSN, Yahoo and Best Buy, for example.

- **SQL injection** is a technique used to take advantage of nonvalidated input vulnerabilities to pass SQL commands through a web application for execution by a back-end database. Attackers take advantage of the fact that programmers often chain together SQL commands with user-provided parameters and, therefore, can embed SQL commands inside these parameters. The result is that the attacker can execute arbitrary SQL queries and/or commands on the back-end database server through the web application.

- **Data exfiltration** is the unauthorized copying, transfer or retrieval of data from a computer or server. Data exfiltration is a malicious activity performed through various techniques, typically by cybercriminals over the Internet or other network. Data exfiltration is also known as data exusion, data exportation or data theft.

- **Staging server** is a server that enables assembling, deploying and testing a software or web site on a server instance, similar to the production server. Typically, software or a web site is deployed on the staging server from the development server when development is complete. A staging server helps to identify the software or web site behavior, experience and performance as it will be visible on the production server. This helps software developers or quality assurance (QA) staff identify and resolve any problems, bugs, performance or usability issues, or other issues before the software or web site is deployed on the production server. The staging server can be a staging database server, staging web site server or staging application server, for example.

Analyzing database configuration is critical to determine vulnerabilities and assure the standard auditing. Database auditing includes:
1. Finding sensitive data and privileges
2. Preventing data access
3. Validating that detection and alert mechanisms are in place
There are multiple mechanisms available that must be in place when databases are configured, including:

- **Data redaction** provides selective, on-the-fly redaction of sensitive data in SQL query results, prior to application display, so that unauthorized users cannot view the sensitive data. It enables consistent redaction of database columns across application modules accessing the same database information. Data redaction minimizes changes to applications because it does not alter actual data in internal database buffers, caches or storage, and it preserves the original data type and formatting when transformed data are returned to the application. Data redaction has no impact on database operational activities, such as backup and restore, upgrade, and patch, and on high-availability clusters.

- **Data masking** obfuscates sensitive data by replacing them with other data—typically characters that will meet the requirements of a system designed to test or still work with the masked results. Masking ensures that vital parts of personally identifiable information (PII), such as the first five digits of a social security number, are obscured or otherwise deidentified.

- **Data encryption** involves converting and transforming data into scrambled, often unreadable, ciphertext using nonreadable mathematical calculations and algorithms. Restoring the message requires a corresponding decryption algorithm and the original encryption key.

### DATABASE AUDITING STEPS

The following steps need to be followed for database auditing.

#### Step 1: Determine if Default Accounts Have Been Changed or Disabled

Default privileged Oracle accounts continue to be the highest risk issue commonly encountered. It is an easy issue to fix and prevent. After installation, Oracle has a number of default accounts, each with a default preset value. Following database install, the Oracle database configuration assistant (DBCA) automatically locks and expires the majority of the default database user accounts. Additionally, DBCA changes the SYSTEM account to the value specified during the installation routine.

If an Oracle database is manually installed, the DBCA never executes, and those dangerous default privileged accounts are never locked and expired. By default, their password is the same as their username. These will be the first credentials that a hacker will attempt to use to connect to the database. As a best practice, each of these accounts should be configured with a strong unique password, and if an account is not required, it should be locked and expired.

To change the password, the following SQL code should be executed:

```
sqlplus> connect mydba
sqlplus> alter user SYSTEM account lock and expire
```

The following SQL can be used to lock and expire those default accounts:

```
sqlplus> connect mydba
sqlplus> alter user SYSTEM account lock and expire
```

The default accounts installed with Oracle vary by version. Figure 1 provides a quick reference of the accounts that are installed by default (if the DBCA is never executed) in Oracle 9, 10 and 11 in an open state.

<table>
<thead>
<tr>
<th>Oracle 9i</th>
<th>Oracle 10g 2</th>
<th>Oracle 11g</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM</td>
<td>SYSTEM</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
</tr>
<tr>
<td>SCOTT</td>
<td>SYSTEMAN</td>
<td>MGMT_VIEW</td>
</tr>
<tr>
<td>DBSNMP</td>
<td>DBSNMP</td>
<td></td>
</tr>
</tbody>
</table>

Source: Muhammad Mushfiqur Rahman. Reprinted with permission.

Starting with Oracle version 11g, database administrators (DBAs) can easily locate any accounts with default passwords (same as username) by using the database view DBA_ USERS_WITH_DEF_PWD.

#### Step 2: Audit the Strength of Oracle Database SID

The Oracle system ID (SID) is a unique value that is required for all clients to connect to the Oracle database. Because it must be unique, there cannot be more than one database with the same SID on the same Oracle server.

If a client connection uses an incorrect SID to connect to an Oracle database, it will get the message “ORA-12505: TNS:listener does not currently know of SID given in connect descriptor.” However, SIDs can be brute forced. There are numerous tools to brute force the Oracle SID, including...
Metasploit modules, operational acceptance testing (OAT) and SIDGuess.

The key to thwarting SID brute-force attacks is to select a SID that is strong. When creating an Oracle SID, the selection should:

- Not be a dictionary word
- Be at least 10 characters in length
- Include at least one special character

Incorporating these elements ensures that the SID is strong, i.e., difficult for an attacker to brute force.

Why does a strong SID matter when the SID is stored as a cleartext value within the Oracle client configuration file, TNSNAME.ora, on every single system that is configured to connect to the database? It matters because as long as an attacker can compromise at least one system that is configured to connect to the Oracle database, obtaining the SID from the TNSNAMES.ORA file is trivial. However, it is important to consider instances where the attacker is external to the organization and has compromised a single host that does not have an Oracle client connection configured. A strong SID will not in and of itself prevent hackers from gaining a connection to an organization’s Oracle database, but it is a good practice as part of a defense-in-depth approach to security.

Step 3: Audit the Oracle Critical Patch Updates
This is one of those security best practice recommendations with which most organizations commonly struggle. Depending on the database schema, Oracle critical patch updates (CPUs) can have significant impact on the Oracle database—significant enough that the organization might have to perform extensive regression testing to ensure that applying the latest Oracle CPUs has no impact on database functionality.

Oracle releases CPUs quarterly on the Tuesday closest to the 17th day of the month. Oracle has a special bulletin page that describes all of the most recent Oracle Critical Patch Updates and Advisories. Fortunately, CPUs are cumulative in nature. One can simply install the latest Oracle CPU to gain all of the security patches since the product’s initial release.

The key to an effective CPU patch process is creating a regimented regression testing process that corresponds to Oracle’s four scheduled releases every year. Even in organizations with the most stringent regression testing processes, the CPUs can usually be architected in such a manner that they can be applied no more than three months after the last CPU release. Additionally, all DBAs should register with the Oracle email Security Alert Advisory Service to ensure timely notification of Oracle patches and security alerts.

There is also a mechanism that Oracle employs if a critical vulnerability is discovered that warrants immediate patch release. Oracle refers to patches released immediately under this program as “off-schedule security alerts.” Since the CPU program began in 2005, there have only been a few times when Oracle released patches under this emergency process. Organizations should develop a method for applying these emergency released patches, but given their historic low volume, the focus should be on the routine applying of CPU patches every quarter.

Step 4: Audit PUBLIC Role for Identification of Unnecessary Privileges
In Oracle, extended routines exist that allow minimally privileged users to execute functions that they otherwise would not be able to execute. These extended routines are called packages, and are roughly equivalent to Extended Stored Procedures in Microsoft SQL Server. A special role, called PUBLIC, acts as a default role assigned to every user in the Oracle database. Any database user can execute privileges that are granted to PUBLIC. This is commonly exploited for database privilege escalation.

These packages and subtypes should be revoked from PUBLIC and made executable for an application only when absolutely necessary.

Step 5: Check That Database Auditing Is Enabled
To identify the malicious or authorized activities in a database, it is important to check that database auditing options are enabled. To ensure that database auditing is enabled, one needs to perform the following activities during the database audit:

- **Audit SYS operations**—By default, Oracle databases do not audit SQL commands executed by the privileged SYS and users connecting with SYSDBA or SYSOPER privileges. If a database is hacked, these privileges are going to be the hacker’s first target. Fortunately, auditing SQL commands of these privileged users is very simple: sqlplus> alter system set audit_sys_operations=true scope=spfile.
- **Enable database auditing**—Again, by default, Oracle auditing of SQL commands is not enabled. Auditing should be turned on for all SQL commands. Database auditing is
turned on with the audit_trail parameter: sqlplus> alter system set audit_trail=DB, EXTENDED scope=spfile.
(Note: The command enables auditing from the database, but not the database vault information, into the table SYS.AUD$.) There are actually four database auditing types: OS, DB, EXTENDED and XML.

• Enable auditing on important database objects—Once auditing has been enabled, it can be turned on for objects where an audit trail is important.

Step 6: Audit to Ensure That Database Triggers for Schema Auditing and Logon/Logoff Events Are Configured
To effectively audit schema changes and logon and logoff events, Oracle provides Data Definition Language (DDL) triggers to audit all schema changes and can report the exact change, when it was made, and by which user.

• Logon trigger—By using a logon trigger, one can send logon and logoff events in real time to another system. Think of it as a syslog daemon for your database events. The following example would send all logon and logoff events to a web server in real-time.

  SQL> create or replace trigger sec_logon after logon on database.

• DDL trigger—Using the DDL triggers, an Oracle DBA can automatically track all changes to the database, including changes to tables, indexes and constraints. The data from this trigger are especially useful for change control for the Oracle DBA. The following example sends events for GRANT, ALTER, CREATE, DROP.

  Command (as user SYS):
  SQL> create or replace trigger DDLTrigger
  AFTER DDL ON DATABASE
  DECLARE
  rc VARCHAR(4096);
  BEGIN
  rc:=utl_http.request('http://192.168.2.201/user'||ora_login_user||';
  DDL_TYPE='||ora_sysevent||';DDL_OWNER='||ora_dict_obj_owner||';DDL_NAME='||ora_dict_obj_name||';sysdate='||to_char(sysdate,'YYYY-MM-DD hh24:mi:ss');
  exception
  when utl_http.REQUEST_FAILED then null; end;
  END;
  / 

• Error trigger—Error triggers are Oracle error messages. They can be useful for detecting attacks from SQL injection and other attack methods.

  For example, command (as user SYS):
  SQL> CREATE OR REPLACE TRIGGER after_error
  AFTER SERVERERROR ON DATABASE
  DECLARE pragma autonomous_transaction; id NUMBER;
  sql_text ORA_NAME_LIST_T; v_stmt CLOB; n NUMBER;
  BEGIN
  n := ora_sql_txt(sql_text);
  IF n >= 1 THEN
  FOR i IN 1..n LOOP
  v_stmt := v_stmt || sql_text(i);
  END LOOP;
  END IF;
  FOR n IN 1..ora_server_error_depth LOOP
  IF ora_server_error(n) in ('900', '906', '907', '911', '917', '920', '923', '933', '970', '1031', '1476', '1719', '1722', '1742', '1756', '1789', '1790', '24247', '29257', '29540') THEN
  INSERT INTO system.oraerror VALUES (SYS_GUID(), sysdate, ora_login_user, ora_client_ip_address, ora_server_error(n), ora_server_error_msg(n), v_stmt);
  END IF;
  END LOOP;
  END after_error; /

Step 7: Audit to Ensure That a DAM Solution Is Implemented
If an organization can afford the extra expense of an additional software product, a database monitoring solution can be very useful. It solves the issue of not being able to monitor the DBA's activity at an organizational level. It also provides useful insight into dangerous SQL queries and role modifications that might indicate an attacker has compromised a database. The key to all database activity monitoring (DAM) solutions is that they operate within memory of the Oracle server and operate independently of the database's native auditing and logging functions. For anyone familiar with network intrusion detection systems (IDSs), DAMs have an analogous function: They operate within the database layer on the server rather than at any of the network layers.
Step 8: Audit to Ensure That Password Management for All Oracle Logins Is Enabled

Oracle provides fairly robust password management for Oracle logins. Unfortunately, none of these are applied in the default Oracle account profile.

In Oracle, logins are assigned an account policy through an Oracle profile. Every login can be applied to only one Oracle profile. If no Oracle profile is specified when the login is created, it is assigned the default Oracle profile.

Oracle covers the syntax for Oracle profiles well, but here are the recommended settings at a high level:

- **Creating profiles**—Oracle profiles are created with:
  ```sql
  CREATE PROFILE profilename LIMIT SQL statement
  ```

  Users are added to the profile with:
  ```sql
  ALTER USER login(s) PROFILE profilename
  ```

- **Account lockout configuration**—Account lockout configuration should be enabled. Locking accounts for 30 days after five invalid attempts greatly reduces the risk of brute-force attacks. If 30 days is not feasible, even a setting of one day greatly reduces the risk of brute-force attacks.

  The following two parameters are used to specify account lockouts in an Oracle profile:
  ```sql
  FAILED_LOGIN_ATTEMPTS 5
  PASSWORD_LOCK_TIME 30
  ```

- **Password expiration**—By expiring passwords, one can help ensure that they are being changed on a periodic basis. Expiring user passwords at least every 90 days is a security best practice. The following parameter is used to specify the number of days that can lapse before a user must change their password:
  ```sql
  PASSWORD_LIFE_TIME 90
  ```

- **Password history**—Without password history, users will most likely use the same password each time they change it. To ensure that users do not reuse passwords, there are two parameters. The important thing to note is that these settings are cumulative and both thresholds must match before users are able to change their password. In general, a password reuse allowance of one time is sufficient in conjunction with a password reuse maximum allowance of 10 or more. Setting the password reuse allowance higher than one time may be problematic if users frequently change their password. The important thing to note with these two settings is that they both should not be set to unlimited:
  ```sql
  PASSWORD_REUSE_TIME 1
  PASSWORD_REUSE_MAX 10
  ```

- **Password complexity verification**—Without a password complexity verification function, users most likely choose simple dictionary words that are easy to remember and easy for a hacker to guess. In Oracle, a user Procedural Language (PL)/SQL script must be set to check the complexity of a user's password. In general, the password verification function should ensure that users’ passwords incorporate the following criteria:
  - Differ from their username
  - Are not a dictionary word
  - Are at least 10 characters in length
  - Include at least one alpha, one numeric and one special character

Step 9: Check to Ensure That Regular Database Security Assessments Are Performed

Every secure configuration that has been discussed could be easily detected with an automated database vulnerability tool. Automated database vulnerability tools provide an excellent way to quickly validate an organization’s Oracle secure configurations. Obviously, these kinds of tools are only useful if one has privileges. They are intended for DBAs, auditors and security professionals to run for regular assessments. These tools are prone to false-positives and, unfortunately, false-negatives, but their benefits greatly outweigh their risk.

Step 10: Determine That Database Traffic Is Encrypted

This recommendation is rarely implemented, except in the most secure organizations. Oracle supports network-level encryption by both Secure Sockets Layer (SSL), using X.509v3 signed certificates, and native encryption without certificates.

The takeaway with network-level encryption is not only that sensitive data in transit are protected when encryption
is employed, but also that the SID is protected. Without encryption, the SID can be easily enumerated through man-in-the-middle attacks.

Step 11: Audit Security Threats and Countermeasures Properly
An organization should create a written security policy to enumerate the security threats it is trying to guard against and the specific measures the organization must take. Security threats can be addressed with different types of measures:

- **Procedural**, such as requiring data center employees to display security badges
- **Physical**, such as securing computers in restricted-access facilities
- **Technical**, such as implementing strong authentication requirements for critical business systems
- **Personnel-related**, such as performing background checks or vetting key personnel

GUIDELINES FOR SECURING USER ACCOUNTS AND PRIVILEGES
Follow these guidelines to secure user accounts and privileges:

1. Practice the principle of least privilege. Oracle recommends granting necessary privileges only. Do not provide database users or roles more privileges than necessary. (If possible, grant privileges to roles, not users.) In other words, the principle of least privilege is that users be given only those privileges that are actually required to efficiently perform their jobs. To implement this principle, restrict the following as much as possible:
   - The number of SYSTEM and OBJECT privileges granted to database users
   - The number of people who are allowed to make SYS-privileged connections to the database
   - The number of users who are granted the ANY privileges, such as the DROP ANY TABLE privilege. For example, there is generally no need to grant CREATE ANY TABLE privileges to a non-DBA-privileged user.
   - The number of users who are allowed to perform actions that create, modify or drop database objects, such as the TRUNCATE TABLE, DELETE TABLE, DROP TABLE statements, and so on
   - The CREATE ANY JOB, BECOME USER, EXP_FULL_DATABASE, and IMP_FULL_DATABASE privileges
   - Library-related privileges to trusted users only
   - Synonym-related privileges to trusted users only
   - Nonadministrative user access to objects owned by the SYS schema
   - Permissions on run-time facilities
2. Lock and expire default (predefined) user accounts.
3. Monitor the granting of the following privileges to only users and roles that need these privileges. By default, Oracle Database audits the following privileges:
   - ALTER SYSTEM
   - AUDIT SYSTEM
   - CREATE EXTERNAL JOB
4. Revoke access as follows:
   - Grant privileges only to roles. Granting privileges to roles and not individual users makes the management and tracking of privileges much easier.
   - Limit the proxy account (for proxy authorization) privileges to CREATE SESSION only.
   - Use secure application roles to protect roles that are enabled by application code.
   - Discourage users from using the NOLOGGING clause in SQL statements.

CONCLUSION
Data are a very decisive resource for any business due to shielding; regularly auditing the database should never be left to chance or patchwork solutions. During the audit period, stakeholders need to identify that a system is configured as per the standard that ensures the mitigation of the data risk.

A complete, all-inclusive auditing solution must be implemented that can easily accomplish each of the following:

- Access and authentication auditing
- User and administrator auditing
- Suspicious activity auditing
- Vulnerability and threat auditing
- Change auditing

Without an all-encompassing auditing solution, organizations put precious data at risk. Corrupt, inaccurate or compromised data equal lost revenue, lost time, and compromised customer and employee relationships.

Auditing is a continuous and ongoing process no matter what system or provider is in use. Even the basics should
be reviewed periodically to avoid a false sense of security. The database is a sensitive component in business; thus, it is important to ensure the database is configured properly to ensure the security of business data.

ENDNOTES


