Database Auditing

Overview:
This document supplements and summarizes the reading related to Database auditing from the Week 7 reading. Regular audits are a critical component of database security

Important: Be sure to enable Audit Trails by setting the parameter values to DB,EXTENDED in the AWS RDS Parameter Group before completing this exercise. Instructions are provided in this week's content area on how to do this.

Background:
Auditing includes monitoring and recording actions from users, applications and other interactions to ensure the security of the applications, databases and IT systems in general. It is different than financial or accounting audits. It also is not about testing of controls but does include implementation and documentation of that implementation as well as risk management. Auditing includes monitoring, recording and analyzing user activity as it pertains to your database.

Auditing generates logs and additional table. So even after determining what rule and activities you need to audit, monitoring the logs and table events are key elements associated with auditing.

Auditing is used to help with compliance and alignment both U.S and International regulations and standards. Several standards are mentioned in the readings including Sarbanes-Oxley Act, Health Insurance Portability and Accountability Act (HIPAA) and the Payment Card Industry (PCI).

PCI is particular useful to help protect credit card transactions as it maintains, evolves and promotes credit card safety for cardholder data. The PCI provides requirements for what types of data can be stored from a customer and if it needs to be encrypted. For example, the primary account number on your credit card can be stored and it needs to be encrypted. The card holder name and expiration can also be stored but it does not need to be encrypted. The CID/CVC2 data cannot be stored. These details are important to know when designing a database system that needs to store credit card data.

Failure to maintain or operate the system as required may lead to mistakes and the accidental disclosure of information, and/or unauthorized charges. Design your auditing strategy to collect the amount of information that you need to meet compliance requirements and those that are of the greatest security concern. For example, auditing every database table in the database is not practical as the performance would be unacceptable and the data collected would be arduous to analyze. However; auditing tables with columns that contain sensitive data, such as salaries, social security numbers or credit card numbers may be worthwhile.

DB auditing:

Auditing a database typically includes monitoring and recording database actions from database and non-database users (e.g. Database and Mobile Applications). Auditing is performed on individual actions via SQL queries. Of interest to auditors are both successful and unsuccessful activities. For example, the fact that a user attempts to use DBA-related privileges multiple times and failed may be just as interesting as the fact that a DBA successfully deleted a log file.

A comprehensive audit will include multiple checks including reviewing expired and unexpired default accounts, timeline of applying patches, database and third party services, password policy and password

policy updates, account lockout policy and others. Often database SQL triggers and procedures are used to capture the data in logs and other tables for future analysis or automatic alerts and notifications.

Security concerns an auditor or other security analyst may need to be aware of includes:

- Phishing – fraudulent email method in an attempt to gather personal and financial information from recipients.
- SQL injection – Non-validated input vulnerabilities to pass SQL commands through a web application for execution by a back-end database.
- Data exfiltration - is the unauthorized copying, transfer or retrieval of data from a computer or server
- Excessive and Unused Privileges – remember the principle of least privilege
- Privilege abuse – absolute power corrupts absolutely.
- Malware – can cause/assist in data exfiltration

Database audits often reveal other issues including weak audit trail, theft of back-up data and storage media, database vulnerability due to slow or lack of critical patches being applied, improper handling, labeling and storage of sensitive data such as proprietary and classified data, distributed denial of service attacks and limited security expertise due to lack of education, training or continuing education.

Threat and attack strategies are constantly changing. Security analysts, auditors, database administrators and users must work to be knowledgeable of the latest threats and vulnerabilities and how to defend against them.

Analyzing your database configuration helps to find sensitive data and privileges and prevent unauthorized data access. An auditor helps in validating the presence of detection and alert mechanisms. Data discovery and classification is a precursor and best practice prior to database design. This process assists in controlling access to and hardening the security of databases containing highly-sensitive data.

Data Protection:
Three methods that help protect data include data redaction, data masking and data encryption. Oracle (and most other databases) provides each of these protection methods. Data redaction provides selective, just-in-time removal of sensitive data in SQL query results. This process occurs prior to the results being displayed ensuring unauthorized users cannot view the sensitive data. Similar to redaction, data masking obfuscates sensitive data by replacing them with other characters. For example, the first 12-digits of a credit card number may be masked such as **********2122. Data encryption converts and transforms data into non-readable data using specialized and algorithms. Restoring the message requires a corresponding decryption algorithm and the original encryption key. Clearly, protecting the keys associated with encryption and decryption is a critical component of strong security plan.

Database Security Best Practices:
There are a number of best practices recommended to keep your database secure. For example, if a user does happen to gain access to your system with elevated privileges, much damage could be done. One defense is to make sure existing accounts with possible elevated privileges have the default

passwords changes and those that are not used should be changed, and accounts locked and expired. The following SQL statement can be used to lock and expire the USERNAME account:

```
ALTER USER USERNAME ACCOUNT LOCK and EXPIRE;
```

The System Identifier (SID) should be >=10 characters, include at least one special character and not be present in the data dictionary. The SID is the unique name for your database instance. Keeping the SID different than most others does help mitigate attacks as the SID is required for gaining access and the adversary would need to guess or find the SID to gain access.

Databases, like all other software do have security vulnerabilities and require patching on a regular basis. Hackers are very aware of the vulnerabilities and look for database systems that aren't patched to exploit those vulnerabilities. Oracle provides quarterly database patch set updates. However; you do need to have a valid support contract and license to be able to download and apply those patches. Although the license and support can be expensive, being able to apply patches is a best practice and required in all production systems.

When auditing, it is a best practice to users who have the CREATE ANY privilege and the PUBLIC role. Any database user can execute privileges that are granted to PUBLIC. These privileges are commonly exploited for database privilege escalation.

Auditing in Oracle:
Auditing parameters in Oracle need to be configured for use. For this class, the Audit parameters are set to DB, Extended. For our labs, this means you can view the actual SQL_TEXT associated with the Audit as well as when and who executed the query. The following SQL statement can be run to determine the current Audit parameters:

```
show parameter audit_trail;
```

As shown in figure 1, running this SQL statement from your student account with show DB, Extended results.

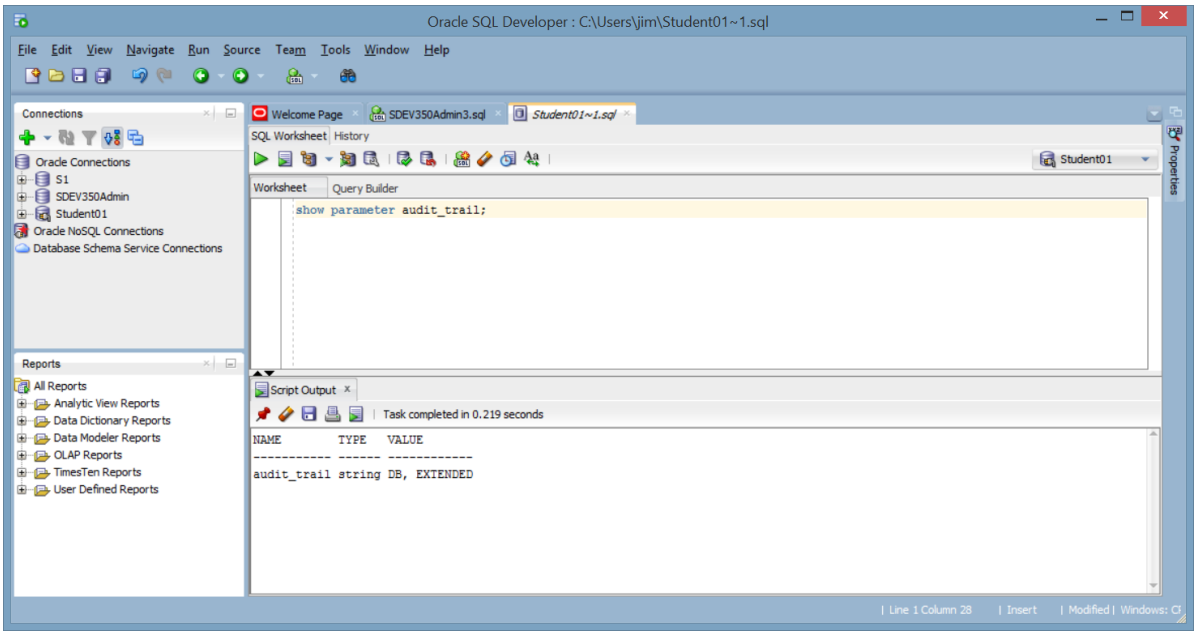

Figure 1 Oracle Audit Parameters

Although other options exist for analyzing and reviewing audit data, this class will focus on the dba_audit_trail and unified_audit_trail data dictionary views. You can use the describe command to show each of the columns associated with these views:

```
desc dba_audit_trail;

desc unified_audit_trail;
```

Although dozens of columns are available to view, popular columns for reports and analysis include

```
DBUSERNAME                      VARCHAR2(30)

EVENT_TIMESTAMP                 TIMESTAMP(6) WITH LOCAL TIME ZONE

ACTION_NAME                     VARCHAR2(64)

OBJECT_NAME                     VARCHAR2(128)

SQL_TEXT                        CLOB
```

For example, the following SQL statement would show all records in the unified_audit_trail for those columns:

```
select dbusername, EVENT_TIMESTAMP,SQL_TEXT,Action_name,Object_name from
unified_audit_trail;
```

AWS RDS doesn't support the Unified Audit Trail for this classroom. So, you will be using the DBA Audit Trails discussed below in most cases.

Running a query on the dba_audit_trail results in the output shown in figure 2.

```
select username, extended_timestamp, action_name,obj_name from
dba_audit_trail;
```

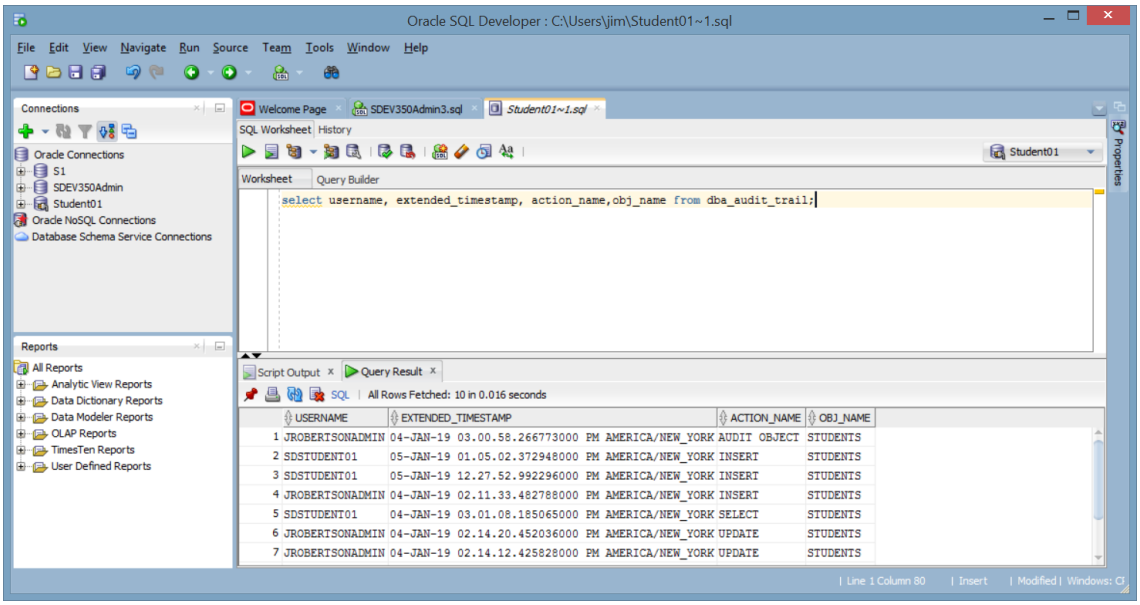

Figure 2 DBA Audit Trail Example

To begin auditing you use the Audit Privilege on Tablename SQL statement. For example, the following will audit Insert and Update statements on the Students table.

```
AUDIT INSERT, UPDATE ON STUDENTS;
```

Once the Audit has been set-up, a query to the dba_audit_trail provides many details. Figure 3, shows some of the data logged into the dba_audit_trail after the Audit has been set-up on the Students table.

JROBERTSON 05-MAY-19
11.49.56.753650000 AM AMERICA/NEW_YORK INSERT STUDENTS

JROBERTSON 05-MAY-19
11.51.01.632340000 AM AMERICA/NEW_YORK UPDATE STUDENTS

Figure 3 Auditing Insert on the Students Table

To stop auditing use the No Audit statement:

```
NO AUDIT Insert, Update on Students;
```

Here are a couple of tips when setting up Audits:

1. Be sure to list the schema name when configuring the Audit (e.g. SDStudent01.Students).
2. Test that your Audit is working by logging in as the user, and then performing the action and finally querying the dba_audit_table
3. There are limitations using AWS Audit as it isn't a true unified_audit_trail. Use the DBA Audit Table for this exercise.

You should experiment and try different Audits and verify the process works as expected. Be sure to use No Audit to stop auditing.