UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Análisis de los principales sistemas de gestión de bases de datos ante ataques básicos

Trabajo Fin de Máster

Presentado por: Armendariz Perez, Iñigo

Director/a: Cuesta Gomez, David

Ciudad: Gatika

Fecha: 28/01/2016

Resumen

La información más sensible de prácticamente la totalidad de las empresas se almacena en sistemas gestores de bases de datos. Estos gestores pueden ser vulnerables a ataques externos, internos o a errores humanos y la información puede verse comprometida y accedida por personas no autorizadas.

Una de las principales tareas del análisis forense es poder realizar un estudio de todo tipo de evidencia digital que esté involucrada en un incidente para que sea admisible en un proceso judicial.

El objetivo principal de este trabajo consiste en realizar una serie de ataques sobre cuatro de los gestores de bases de datos más conocidos (MySQL, PostgreSQL, Microsoft SQL Server y Oracle) y posteriormente analizarlos y ver si somos capaces de responder a las cuatro preguntas clave del análisis forense; que, como, cuando y quien ha realizado el ataque.

Palabras Clave: Análisis Forense, SGBD, vulnerabilidad, ataque

Abstract

The most sensitive information of virtually every company, is kept within database management systems. These systems can be vulnerable to external or internal attacks and also to human errors. In these cases information could be in danger as non allowed individuals might have access to it.

One of the main tasks of the forensic analysis consists in the development of a study based on the available digital evidences involved in an incident, so a trial can be carried out.

The main goal of this task consists in carrying out a series of attacks against four of the most famous database managers (MySQL, PostgreSQL, Microsoft SQL Server y Oracle), to later analyze them and see if they were able to answer the four key questions in the forensic analysis; what, how, when and who carried out the attack.

Keywords: forensic analysis, DBMS, vulnerability, attack.

Índice

Índice de Figuras

Índice de Tablas

1. Introducción

Hoy en día, la información más sensible de la mayor parte de las empresas se encuentra almacenada en bases de datos. Estas bases de datos gestionadas por los sistemas gestores de bases de datos, en adelante SGBD, pueden ser propietarias (pago por licencia) o de software libre.

Los SGBD, al igual que todos los sistemas informáticos, contienen vulnerabilidades que pueden ser aprovechadas por terceras personas no autorizadas para acceder a la información.

Esta información sensible hace que los hacker centren su esfuerzo en acceder a ella por medio de las muchas vulnerabilidades que se encuentran en los SGBD, vulnerabilidades que pueden ser debidos a problemas de seguridad en el software o a una mala configuración por parte del administrador de sistemas.

La vulneración de la información no siempre es debido a ataques externos, en muchos casos puede ocurrir desde dentro de la organización, bien por ataques intencionados o bien por errores humanos.

La informática forense es un proceso de investigación y las evidencias obtenidas de esta investigación tienen que poder ser presentadas como medio de prueba en un procedimiento judicial. Una de sus principales tareas es poder realizar un estudio de todo tipo de evidencia digital que esté involucrada en un incidente.

Cuando la evidencia a investigar se trata de la alteración de información almacenada en una base de datos entra en juego el análisis forense sobre bases de datos.

En este trabajo se va a tratar de analizar cuatro gestores de bases de datos, dos de ellos propietarios (Microsoft SQL Server y Oracle) y dos de software libre (MySQL y PostgreSQL), después de recibir una serie de ataques internos y externos para ver si somos capaces de responder a las cuatro preguntas clave del análisis forense:

- ¿Qué se ha alterado?
- ¿Cómo se ha alterado?
- ¿Quién lo ha alterado?
- ¿Cuándo se ha alterado?

1.1. Objetivos

En este trabajo voy a tratar de realizar un análisis forense sobre bases de datos comerciales y de software libre. Me centrare en los gestores comerciales Microsoft SQL Server y Oracle, y en los gestores de código libre MySQL y PostgreSQL.

Se establecerán cuatro escenarios de prueba donde realizar ataques internos y externos a una base de datos para a continuación realizar un análisis forense y comparar los resultados.

El objetivo no es analizar distintas herramientas de análisis forense, sino **ver la diferencia que existe entre los distintos SGBD para obtener los datos de los ataques realizados**. Quiero comparar la facilidad de obtener los datos y la calidad de los mismos procurando no utilizar herramientas de terceros sino utilizando las propias herramientas de cada gestor.

Los objetivos específicos planteados son los siguientes:

- Definir los sistemas gestores de bases de datos, haciendo hincapié en MySQL, PostgreSQL, Microsoft SQL Server y Oracle, su seguridad e implementación de auditorías.
- Definir los conceptos del análisis forense. Objetivos y etapas.
- Implementar cuatro escenarios virtualizados sobre Microsoft Windows 2012 Server:
    1. MySQL, versión 5.6.17
    2. PostgreSQL, versión 9.4
    3. Microsoft SQL Server, versión 2014
    4. Oracle, versión 12c
- Implementar auditorías en los cuatro escenarios.
- Realizar cuatro ataques sobre cada escenario:
    1. Ataque interno. Usuario administrador que modifica datos.
    2. Ataque interno. Usuario de la empresa que modifica datos.
    3. Ataque externo. SQL Injection.
    4. Ataque externo. Fuerza bruta.
- Realizar un análisis de cada escenario y de cada ataque, obteniendo datos de cada SGBD con y sin auditoría.
- Comparar los resultados de los cuatro escenarios.

El aporte principal del presente trabajo es comparar las trazas de seguridad que dejan las bases de datos más extendidas al ser explotadas, estudio que hasta donde me consta no se ha realizado con anterioridad.

1.2. Estructura del trabajo

La estructura del trabajo es la siguiente:

En el primer capítulo se realiza una introducción al trabajo y a los objetivos del mismo.

En el capítulo 2 se realiza una exposición de los fundamentos y objetivos del análisis forense y del entorno legal a la hora de aplicarlo. También se realiza una exposición sobre la seguridad en los sistemas gestores de base de datos e implementación de auditorías.

En el capítulo 3 se implementan cuatro escenarios con los siguientes sistemas gestores de base de datos: MySQL, PostgreSQL, Microsoft SQL Server y Oracle. A continuación se realizan cuatro ataques (dos internos y dos externos) sobre cada escenario y se realiza un análisis sobre cada uno de ellos para intentar responder a las cuatro preguntas fundamentales del análisis forense, que, como, cuando y quien ha alterado los datos. Y por último se comparan los resultados

En el capítulo 4 se presentan las conclusiones extraídas a los largo del análisis de los cuatro escenarios.

Para finalizar, se recogen las referencias utilizadas a la hora de realizar el trabajo.

2. Estado del arte

2.1. Análisis forense

Para realizar un análisis forense orientado a incidentes en gestores de bases de datos, es necesario entender cómo funciona la informática forense, puesto que es en esta ciencia donde se establecen los procedimientos que permiten investigar y detectar toda evidencia.

Según el Instituto Nacional de Ciberseguridad [1 Incibe], la informática forense es:

"El proceso de investigación de los sistemas de información para detectar toda evidencia que pueda ser presentada como medio de prueba fehaciente para la resolución de un litigo dentro de un procedimiento judicial"

Así pues, la informática forense es un proceso de investigación y las evidencias obtenidas de esta investigación tienen que poder ser presentadas como medio de prueba en un procedimiento judicial.

2.1.1. Evidencia Digital

Una de las principales tareas del análisis forense es poder realizar un estudio de todo tipo de evidencia digital que esté involucrada en un incidente para que sea admisible en un proceso judicial.

Una evidencia digital es todo elemento capaz de almacenar información en un sistema informático o electrónico y que permita constatar o esclarecer un hecho.

La evidencia digital posee, entre otros, los siguientes elementos que la convierten en un constante desafío para aquellos que la analizan para buscar la verdad:

1. Es volátil.
2. Es anónima.
3. Es duplicable.
4. Es alterable y modificable.
5. Es eliminable

Podemos clasificar la evidencia digital en dos grupos:

1. Evidencia No Volátil: Son evidencias almacenadas en discos duros o en otros medios informáticos y que se mantienen después que el ordenador es apagado. Estas evidencias no cambian con facilidad, aunque pueden alterar su valor si no se actúa con cuidado.
2. Evidencia Volátil: Son evidencias que por su naturaleza inestable se pierde al apagarse el ordenador. Estas evidencias cambian con facilidad y se encuentran alojadas temporalmente en la memoria RAM, en la memoria cache, tablas de enrutamiento, procesos en ejecución, etc.

Según Paul Henry [2 Best practices in digital evidence collection] el orden de volatilidad (de mayor a menor) de las evidencias digitales es:

- La caché de la CPU (No se podrá capturar).
- La tabla de enrutamiento, la caché ARP, la lista de usuarios que han iniciado sesión en el equipo, la lista de procesos y en general cualquier otra evidencia volátil que pueda desaparecer al apagar el equipo.
- La memoria RAM.
- Los archivos temporales y el espacio de intercambio.
- Los datos del disco duro.
- Datos almacenados remotamente.
- Datos almacenados en dispositivos removibles.

2.1.2. Objetivos del análisis forense

El principal objetivo de un análisis forense es el esclarecer los hechos ocurridos en base a una serie de evidencias recogidas en la escena del crimen. Se deben averiguar los siguientes puntos:

- Que se ha alterado: Se deben detectar los accesos o cambios no autorizados, tanto si han ocurrido modificaciones o eliminaciones, como si han ocurrido lecturas o copias.
- Como se ha alterado. Se deben reconstruir los pasos dados por el autor para comprender como ha sido posible el acceso o cambio.
- Quien lo ha alterado: Intentar detectar quien ha sido el autor material de los hechos, o desde que maquina se ha producido.

2.1.3. Etapas de un análisis forense

Según la metodología del NIST [3 National Institute of Standards and Technology], las etapas de un análisis forense se dividen en:

Figura 1. Etapas análisis forense. Extraída de "Apuntes del profesor. Análisis Forense"

- Recolectar: La recolección de evidencias es el primer paso del análisis forense. Consiste en aislar la escena del crimen y recolectar las evidencias sin modificarlas si es posible, si no se documentarán los cambios. Se duplican o clonan los dispositivos implicados para un posterior análisis. En esta fase habrá que tener mucho cuidado en la adquisición de los datos puesto que cabe la posibilidad de incumplir los derechos fundamentales del atacante
- Preservar: La preservación garantiza que las evidencias recolectadas son las que se analizarán, cumpliendo con la cadena de custodia. Consiste en guardar las evidencias y evitar que se alteren
- Analizar: Se realiza un estudio con los datos adquiridos en la fase de recolección, se analizan las evidencias adquiridas. Se tendrá que averiguar que se ha alterado, como se ha alterado y quien lo ha alterado.
- Presentar: Presentar los resultados en un informe con la descripción del equipo objeto de nuestra investigación, los procesos empleados y sus resultados, y las conclusiones

2.1.4. Adquisición de Evidencias

Antes de realizar la adquisición de evidencias será necesario acotar la escena del crimen, para ello se deberá tener en cuenta que equipos han sufrido el incidente de manera directa y que otros equipos pueden estar relacionados.

A continuación habrá que plantearse si es viable una adquisición en caliente de las evidencias, si se tiene que realizar el análisis en caliente de las mismas o si es posible llevarse las evidencias y analizarlas en frio.

La adquisición de una evidencia se realiza creando un archivo que contiene toda la información contenida en la evidencia original, incluyendo los espacios marcados como vacios o libres si se trata de un disco duro o similar.

Adquisición de evidencias volátiles

Las evidencias volátiles será necesario adquirirlas en caliente, es decir, con el equipo encendido, puesto que al apagarlo estas evidencias desaparecen. El principal problema en este tipo de adquisiciones es que las evidencias se degradan e incluso pueden perderse.

Las evidencias volátiles habrá que adquirirlas en el orden descrito anteriormente en el punto 2.1.1 y se utilizarán herramientas específicas en función de lo que se pretenda adquirir. Las herramientas a utilizar tendrán que ser independientes del equipo a examinar

Adquisición de evidencias no volátiles

A diferencia de las evidencias volátiles, las no volátiles se pueden adquirir tanto en frio (equipo apagado) como en caliente (equipo encendido).

El decantarnos por una adquisición en frio o en caliente dependerá de los siguientes factores:

- Que el equipo pertenezca a un sistema crítico y no pueda apagarse.
- Que el dispositivo se encuentre cifrado, lo cual nos impedirá la posibilidad de apagarlo porque al encenderlo de nuevo no tendremos las claves de cifrado.
- Que el equipo se encuentre expuesto a ataques desde el exterior. Si es así puede que las evidencias se alteren en una adquisición en caliente.

En el documento RFC 32272 [7 Directrices para la recopilación de evidencias y su almacenamiento], tal y como indica su titulo, se recogen las directrices para la recopilación de las evidencias y su posterior almacenamiento.
Dichas directrices a seguir y la toma de evidencias se documentan también en la página web de INCIBE [6 Guía de toma de evidencias]

2.1.5. Preservación de la integridad e identidad de las evidencias

Una vez adquiridas las evidencias, debemos protegerlas con el fin de evitar la modificación de las mismas. Se debe garantizar la no alteración de la evidencia original, asegurar que lo que se está analizando es exactamente lo mismo que lo contenido en la evidencia digital.

Para verificar que lo analizado es una copia exacta de la evidencia original, o que la integridad de dicha copia se mantiene a lo largo de todo el proceso de análisis se utilizan las funciones hash.

Una función hash debe reunir las siguientes dos características:

- Para cada entrada, la función hash generara siempre una única salida. Esto implica que cualquier alteración en la entrada, por mínima que sea, alterara la salida.
- A partir de la salida de la función hash es imposible obtener la entrada original.

El proceso de adquisición de las evidencias será el siguiente:

Figura 2. Adquisición de evidencias. Extraída de "Apuntes del profesor. Análisis Forense"

Para asegurarnos de que no se ha alterado la evidencia original y que la copia es exacta, el primer hash a la evidencia original (HA1) tendrá que ser igual al segundo hash de la evidencia original (HA2) e igual al hash de la copia (HB).

2.2. Entorno legal

En este apartado vamos a tratar aquellos aspectos legales que guardan relación con los delitos informáticos, que forman la base de un proceso judicial y que hay que tener en cuenta a la hora de realizar un análisis forense.

El proceso judicial, es el conjunto de actos realizados por un tribunal, desde la demanda hasta la sentencia o decisión judicial.

Podemos hablar de proceso en distintos contextos y con diferentes normativas, así, tenemos:

- Procesos Civiles: Ley de Enjuiciamiento Civil 1/2000. (normativa de aplicación supletoria al resto de los procedimientos). Mediante la Ley de Enjuiciamiento Civil se establece el marco legal en el que se regulan los procesos civiles y los tribunales.
- Procesos Penales: Ley de Enjuiciamiento Criminal.
- Procesos Sociales: Ley reguladora de la Jurisdicción social.
- Procesos Contencioso Administrativos: Ley reguladora de la Jurisdicción Contencioso administrativa.

Todos estos procesos constan de tres fases:

1. Fase de alegaciones: Se exponen los hechos y las razones jurídicas en virtud de las cuales quieren obtener un pronunciamiento judicial favorable a sus intereses.
2. Fase de prueba: Las partes intentarán convencer al tribunal de la existencia o no de un dato procesal determinado. Hay que probar todo lo que se alega, por ejemplo, mediante la aportación un dictamen pericial.
3. Fase de conclusiones: Las partes argumentarán al tribunal las razones por las cuales consideran que han sido probados los hechos alegados.

2.2.1. Derechos fundamentales

La constitución española otorga a los ciudadanos una serie de derechos fundamentales y libertades públicas. Estos derechos están regulados por el título I de la Constitución [8 Constitución Española].

De todos estos derechos son especialmente reseñables los siguientes:

- Derecho a la seguridad jurídica y tutela judicial. Este derecho garantiza un proceso penal con garantías.
- Derecho al secreto de las telecomunicaciones.

- Derecho a la vida privada. Este derecho incluye el derecho a la intimidad, a la vida privada y al honor. También se incluye la limitación del uso de la informática para proteger la intimidad.
- Derecho a la protección de datos.

2.2.2. Ley de Protección de Datos de Carácter Personal

Esta ley protege a las personas físicas en todo lo relativo al tratamiento que se pueda realizar de los datos referentes a su persona, ya sean privados o públicos. [10 Ley de Protección de Datos de Carácter Personal]

Su objetivo principal es controlar quien tiene los datos, para que se usan y a quien los cede. Se consideran datos de carácter personal cualquier tipo de información de personas físicas que las identifique o las hagas identificables.
Se detallan tres niveles de seguridad:

- Básico: Todos los sistemas con datos personales.
- Medio: Todos los datos del nivel básico, además de los datos sobre comisión de infracciones administrativas o penales, hacienda pública, servicios financieros, solvencia patrimonial y crédito.
- Alto: Todos los datos del nivel medio, además de los datos sobre ideología, religión, creencias, origen racial, salud o vida sexual.

2.2.3. Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico

La LSSI regula y protege todas las actividades que se realizan por medios electrónicos y tengan carácter comercial o persigan un fin económico. [11 Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico]

2.2.4. Ley de conservación de datos relativos a las comunicaciones y las redes públicas

Su objetivo es el de conservar todos los datos que puedan ser relevantes para rastrear las actividades ilícitas. Se establece una regulación para los operadores de telecomunicaciones para retener determinados datos y que estos estén a disposición de los cuerpos de

seguridad del estado. [12 Ley de Conservación de Datos Relativos a las Comunicaciones y las Redes Pública]

Los datos a conservar se dividen en tres grupos:

- Telefonía fija
- Telefonía móvil
- Acceso a Internet, correo electrónico y telefonía por Internet

2.2.5. Código penal

En el código penal se establecen todas las actividades que se tipifican como delito. La definición de delito viene descrita en el artículo 10 del Código penal (Ley Orgánica 10/1995, de 23 de noviembre) (CP): "son delitos o faltas las acciones y omisiones dolosas o imprudentes penadas por la Ley."

Son varios los artículos del código penal que directa o indirectamente se refieren a las acciones cometidas a través de medios informáticos. Se citan a continuación aquellos delitos del Código Penal que guardan relación con los delitos informáticos. [13 Código Penal]

Delitos contra la propiedad intelectual o la propiedad industrial

- En relación a la propiedad intelectual, se castigará a quienes reproduzcan, distribuyan o comuniquen públicamente, parte o totalidad de una obra literaria, artística o científica, con ánimo de lucro y en perjuicio de terceros (CP arts. 270 a 272).
- Respecto a la propiedad industrial, no sólo es delito la copia directa (CP arts. 273 a 277) sino también comerciar con copias ilegales de productos protegidos (CP arts. 294 a 308).

Delitos contra el derecho a la intimidad

- Interceptación de comunicaciones utilizando la informática como medio, para el descubrimiento, revelación de secretos y otras agresiones a la intimidad. (CP arts. 197 a 201). Si además del delito contra la intimidad, los datos personales están protegido, se sumaría una sanción por la infracción de la LOPD.

Delitos contra el patrimonio

- En los delitos contra el patrimonio, se consideran los actos de estafas, apropiación indebida (CP arts. 252 a 254) y fraudes (CP arts. 255 y 256)

Delitos de sabotaje informático

- En este caso, la informática no sería el medio del delito, si no el objeto de este, estando ante un delito de daños y estragos. El código penal regula estos daños y/o estragos en los artículos 263 a 269, 346,376, 351 y siguientes.

Delitos contra la libertad y amenazas

- Efectuar amenazas sirviéndose de la informática o la telemática, por ejemplo a través de redes sociales o mail (CP arts. 169 y siguientes).

Delitos contra el mercado y los consumidores

- Revelación de secretos (CP art. 278).
- Publicidad engañosa (CP art. 282).
- Falsedades documentales (CP arts. 390 y siguientes).

2.3. Bases de datos relacionales

El modelo relacional fue introducido en la década de 1970 por E. F. Codd de IBM y representó un importante avance para usuarios y diseñadores.

Una base de datos es un conjunto de datos organizados y estructurados según un determinado modelo de información y refleja los datos y las relaciones que existen entre ellos.

Podemos definir una base de datos relacional como una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas desde las cuales se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base.

La interfaz estándar de acceso a una base de datos relacional es el lenguaje de consultas estructuradas SQL.

El conjunto de tablas que forman la base de datos relacional contienen datos provistos en categorías predefinidas. Cada tabla contiene una o más categorías de datos en columnas y cada fila contiene una instancia única de datos para las categorías definidas por las columnas.

2.3.1. Sistema de gestión de base de datos relacionales

Un sistema de gestión de bases de datos relacionales RDBMS (Relational Database Management System) es un programa que permite crear, actualizar y administrar una base de datos relacional. Generalmente, los sistemas de gestión de bases de datos relacionales utilizan el lenguaje de consultas estructuradas SQL para acceder a la base de datos.

Existen multitud de sistemas de gestión de bases de datos relacionales, tanto comerciales como libres

- RDBMS comerciales:
  - Oracle
  - Microsoft SQL Server
  - Informix
  - FoxPro
  - DB2
  - dBase

- Etc.

- RDBMS libres:
  - MySQL
  - PostgreSQL
  - Firebird
  - Sqlite
  - Etc.

	Plataforma			
SGBD	Windows	Linux	Mac OSX	BSD
Oracle	Sí	Sí	Sí	Sí
SQL Server	Sí	No	No	No
MySQL	Sí	Sí	Sí	Sí
PostgreSQL	Sí	Sí	Sí	Sí

Tabla 1. Soporte de Sistema Operativo

La iniciativa DB-Engines [30 Ranking of Relational DBMS], creada y mantenida por Solid IT, reúne información sobre los distintos sistemas de gestión de bases de datos. Entre otras cuestiones, muestra un ranking que se actualiza mensualmente clasificando los sistemas de gestión de base de datos de acuerdo a su popularidad. En la tabla siguiente se puede observar los 10 SGBD más populares en base a una serie de parámetros:

- Número de menciones del sistema en los sitios web.
- Interés general en el sistema.
- Frecuencia de las discusiones técnicas sobre el sistema.
- Número de ofertas de empleo en el que se menciona el sistema.
- Número de perfiles en redes profesionales en los que se menciona el sistema.
- Relevancia en las redes sociales.

Rank Dec 2015	Rank Nov 2015	Rank Dec 2014	DBMS	Database Model	Score Dec 2015	Score Nov 2015	Score Dec 2014
1.	1.	1.	Oracle	Relational DBMS	1497.55	+16.61	+37.76
2.	2.	2.	MySQL	Relational DBMS	1298.54	+11.70	+29.96
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1123.16	+0.83	-76.89
4.	4.	4.	PostgreSQL	Relational DBMS	280.09	-5.60	+26.09
5.	5.	5.	DB2	Relational DBMS	196.13	-6.40	-14.13
6.	6.	6.	Microsoft Access	Relational DBMS	140.21	-0.75	+0.31
7.	7.	7.	SQLite	Relational DBMS	100.85	-2.60	+6.15
8.	8.	8.	SAP Adaptive Server	Relational DBMS	81.47	-2.24	-4.52
9.	9.	9.	Teradata	Relational DBMS	75.72	-1.37	+8.32
10.	10.	↑ 11.	Hive	Relational DBMS	55.27	+0.36	+18.90

Tabla 2. Ranking SGBD más populares. Extraído de DB-Engines

En su página web existe la posibilidad de comparar distintos SGBD en base a diferentes parámetros (Lenguaje de implementación, DBaaS, SO soportado, Esquema de datos, soporte XML, métodos de partición, etc.). En el siguiente enlace se puede visualizar una comparación entre MS SQL Server, Oracle, MySQL y postgreSQL. [34 System Properties Comparison Microsoft SQL Server vs. MySQL vs. Oracle vs. PostgreSQL]

En enero de 2014, en la web “The High-Performance SQL Blog” realizaron un estudio comparando el rendimiento entre MySQL, PostreSQL, SQL Server y Oracle. Sobre tres consultas distintas obtuvieron los tiempos de respuesta de cada gestor. [36 Count Distinct Compared on Top 4 SQL Databases]
Se puede consultar el estudio en:

El resultado fue el siguiente:

	MySQL	Postgres	SQL Server	Oracle
Query 1	27.5s	348s	4.47s	6.09s
Query 2	25.9s	10.6s	2.13s	3.70s
Query 3	13.3s	7.13s	2.21s	3.71s

Tabla 3. Comparativa SGBD. Extraído de The High-Performance SQL Blog

En otro estudio similar [35 Mysql vs Postgres vs Oracle vs Mssql performance], se comparo el rendimiento sobre 9 operaciones de inserción y selección sobre los cuatro SGBD. En el resultado que se puede observar en la tabla siguiente Oracle fue el de mejor rendimiento seguido de MS SQL Server, PostgrerSQL y MySQL:

test no	test name	mysql result	points	postgres result	points	oracle	points	mssql	points	max
1	Insert 10K persons and 20K addresses (rows/s)	300,00	36,86	370,00	45,45	814,00	100,00	720,00	88,45	814,00
2	update 20K rows (execution time)	0,73	43,84	0,70	45,71	0,32	100,00	0,35	91,43	0,32
3	select by primary key (exec/s)	3052,00	100,00	2739,00	89,74	2994,00	98,10	2673,00	87,58	3052,00
4	select by index (rows/s)	3462,00	84,27	3481,00	84,74	4108,00	100,00	2974,00	72,40	4108,00
5	join by address_type name (exec/s)	0,01	0,00	1769,00	52,48	3371,00	100,00	1688,00	50,07	3371,00
6	join by address_type id (exec/s)	2134,00	54,68	1405,00	36,00	3903,00	100,00	1886,00	48,32	3903,00
7	join and order by person id (exec/s)	0,01	0,00	1393,00	34,35	4055,00	100,00	2163,00	53,34	4055,00
8	full scan (execution time)	1,65	42,42	1,00	69,72	3,60	19,44	0,70	100,00	0,70
9	join,group by, order by (exec/s)	1718,00	100,00	1165,00	67,81	0,01	0,00	1577,00	91,79	1718,00
	sum		**462**		**526**		**718**		**683**	
	percent		**64**		**73**		**100**		**95**	

Tabla 4. Comparación rendimiento SGBD. Extraído de phpdao.com

2.4. MySQL

La arquitectura de MySQL separa el motor de almacenamiento, que se encarga de la entrada-salida y representación de la información en memoria secundaria, del resto de componentes.

La siguiente figura muestra la arquitectura de MySQL. [17 Visión general de la arquitectura de MySQL]

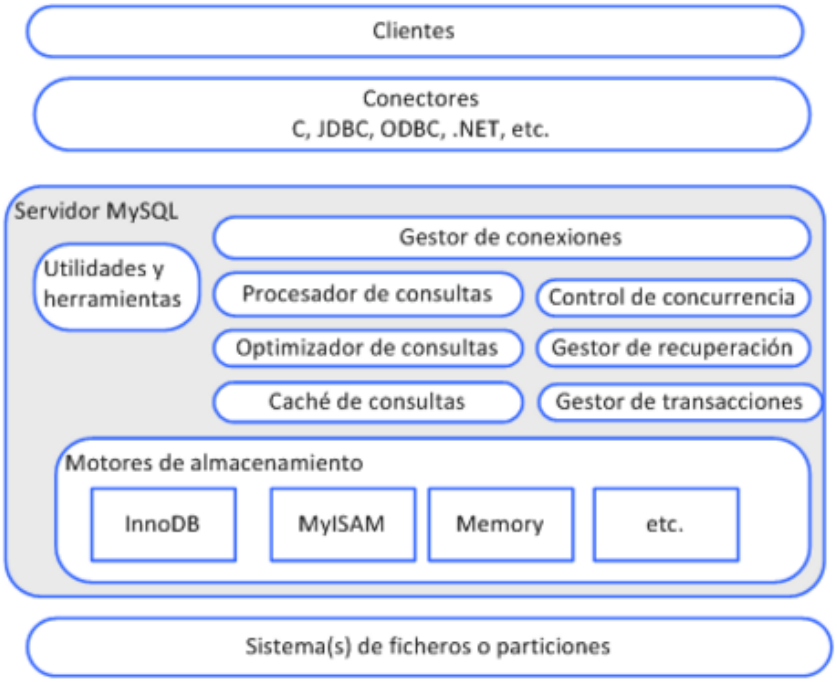


Figura 3. Arquitectura MySQL. Extraída de cnx.org

2.4.1. Motores de almacenamiento

Uno de los elementos más notables de la arquitectura de MySQL son los motores de almacenamiento reemplazables (pluggable storage engine architecture), su función es hacer una interfaz abstracta de funciones comunes de gestión de datos en el nivel físico, de esta manera el gestor de almacenamiento puede intercambiarse. Incluso un mismo servidor MySQL puede utilizar diferentes motores de almacenamiento para diferentes bases de datos.

Un motor de almacenamiento puede implementar los siguientes elementos:

- Concurrencia. La política de bloqueos se implementa en el motor de almacenamiento (o se puede no implementar ninguna). Una estrategia de bloqueos por fila permite una

mayor concurrencia, pero también consume más tiempo de procesamiento en aplicaciones en las que la concurrencia no es realmente grande.

- Soporte de transacciones. No es necesario en todas las aplicaciones.
- Comprobación de la integridad referencial.
- Almacenamiento físico, incluyendo todos los detalles de la representación en disco de la información.
- Soporte de índices. Como la forma y tipo de los índices depende mucho de los detalles del almacenamiento físico, cada motor de almacenamiento proporciona sus propios métodos de indexación.
- Cachés de memoria. MySQL implementa cachés comunes en el gestor de conexiones y la caché de consultas, pero algunos motores de almacenamiento pueden implementar cachés adicionales.
- Otros elementos para ayudar al rendimiento, como puede ser el uso de múltiples hilos para operaciones paralelas o mejoras de rendimiento para la inserción masiva.

2.4.2. Conectores

Los conectores de MySQL son bibliotecas en diferentes lenguajes de programación que permiten conectarse con servidores MYSQL y ejecutar consultas, ya sea de forma remota o local.

2.4.3. Gestor de conexiones

La gestión de conexiones es la responsable de mantener las múltiples conexiones de los clientes. Como las conexiones consumen recursos, este gestor puede configurarse para limitar el número de conexiones concurrentes, y también implementa el pool de conexiones.

2.4.4. Procesamiento y optimización de consultas

Al llegar una consulta al gestor de MySQL, esta se analiza sintácticamente y se toman una serie de decisiones como determinar el orden de lectura de las tablas, el uso de los índices, la reescritura de la consulta más eficientemente, etc.

2.4.5. Cache de consultas

En la cache se guardan consultas y sus resultados. Antes de la optimización, el procesador de consultas buscara la consulta en la cache para evitar realizar el trabajo otra vez.

2.4.6. Control de concurrencia

El control de concurrencia se utiliza para evitar que lecturas o escrituras simultáneas a los mismos datos produzcan inconsistencia o efectos no deseados.

2.4.7. Gestión de transacciones y recuperación

Esta gestión permite declarar una consulta o un conjunto de ellas como una sola transacción. Si alguna de las consultas falla, el servidor anulara el total de la transacción volviendo atrás en las consultas ya ejecutadas.

2.5. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.

Utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de *multihilos* para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará al resto y el sistema continuará funcionando.

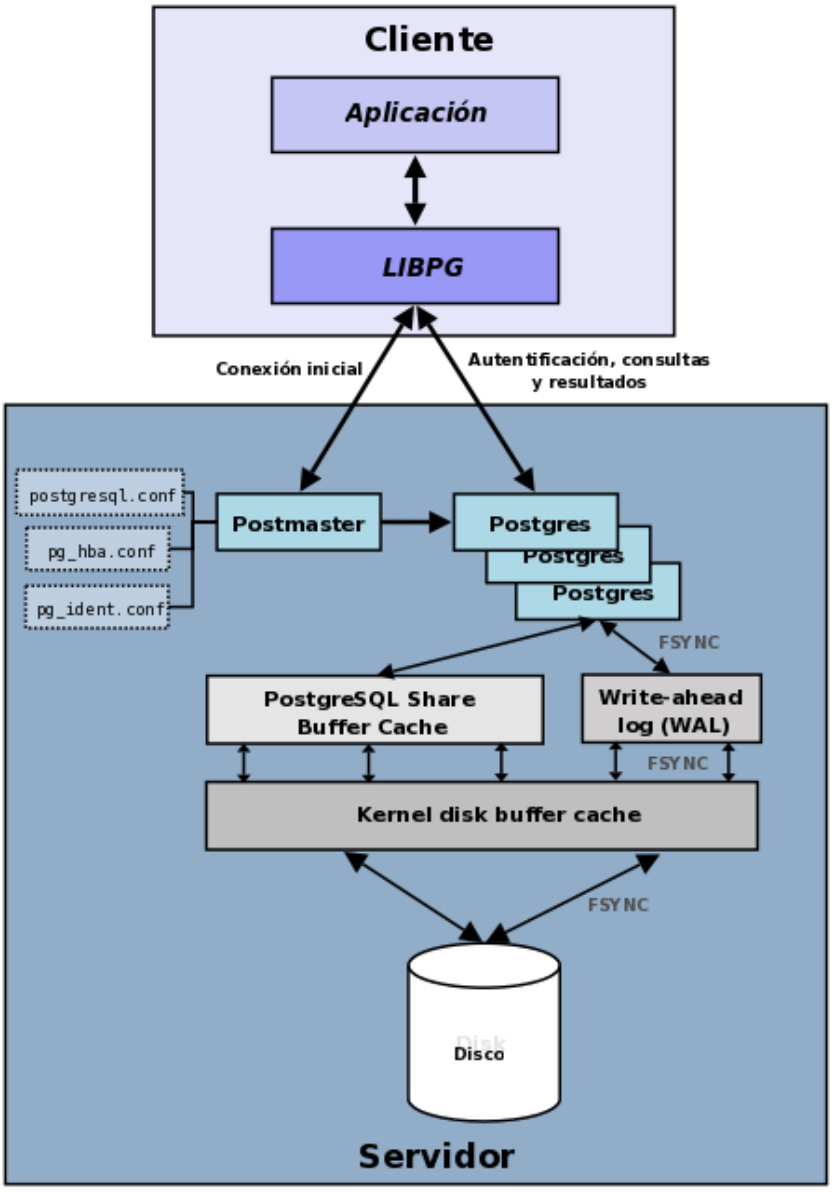


Figura 4. PostgreSQL. Extraído de postgresql.org.es

- **Aplicación cliente**: La aplicación cliente utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir vía TCP/IP ó sockets locales.
- **Demonio postmaster**: Es el proceso principal de PostgreSQL. Se encarga de escuchar por un socket las conexiones entrantes de clientes. También se encarga de crear los procesos hijos que se encargarán de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes
- **Ficheros de configuración**: Los 3 ficheros principales de configuración son postgresql.conf, pg_hba.conf y pg_ident.conf
- **Procesos hijos postgres**: Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes
- **PostgreSQL share buffer cache**: Memoria compartida usada por PostgreSQL para almacenar datos en caché.
- **Write-Ahead Log (WAL)**: Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO)
- **Kernel disk buffer cache**: Caché de disco del sistema operativo
- **Disco**: Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione

2.6. Microsoft SQL Server

SQL Server es la suite de productos Microsoft utilizada para gestionar bases de datos. Dispone de componentes de servidor y herramientas de administración disponibles para ser instaladas dependiendo de la edición que se haya adquirido.

Los cuatro componentes más importantes de SQL Server son:

- Motor de Base de Datos. Es la parte principal de SQL Server. Se encarga de almacenar, procesar y proteger los datos que se ingresan
- Servicios de Integración: Es el responsable de la migración de datos, es decir importar y exportar datos. Cuenta con herramientas para transformar los datos que están siendo movilizados.
- Servicios de Reporte: Aquí se crean, administran e implementan informes tabulares, matriciales y gráficos. En esta plataforma también se pueden desarrollar aplicaciones de informes.
- Servicios de Análisis: Se incluyen herramientas para crear y administrar aplicaciones de procesamiento analítico en línea (OLAP) y minería de datos

2.6.1. Instancias

Una instancia de SQL Server, es un servicio de aplicación auto contenida que implica archivos del sistema operativo, las estructuras de memoria, los procesos de segundo plano y la información de registro. Una instancia está representada por un servicio en Windows y puede estar en ejecución o estado detenido. Cuando se ejecuta, una instancia ocupa una porción de la memoria del servidor, y también genera una serie de procesos de segundo plano.

La parte central de una instancia de SQL Server son sus bases de datos. Una base de datos de SQL Server es el repositorio de datos y el código del programa para la manipulación de esos datos. Si una instancia no se está ejecutando, no se podrá acceder a las bases de datos dentro de ella.

SQL Server consta de bases de datos del sistema y bases de datos de usuario. Cuando se instala una instancia por primera vez, se crean cinco bases de datos del sistema: model, tempdb, master, msdb y resource.

Cada instancia que exista en un equipo contendrá las cinco bases de datos del sistema, y no podrá iniciar si cualquiera de sus bases de datos del sistema (excepto msdb) esta inaccesible o dañada.

Las bases de datos de usuario almacenarán la información de las organizaciones. Una instancia de SQL Server siempre incluye bases de datos (aunque solo sean las del sistema) y una base de datos siempre estará asociada a una única instancia.

2.6.2. Nivel Físico

A nivel físico, una base de datos de SQL Server está representada por un conjunto de archivos del sistema operativo donde resida. Hay dos tipos de archivos de base de datos:

- Archivo de Datos (data file). Es el repositorio central de información en una base de datos SQL
- Archivo de Registro de Transacciones (transaction log file). Registra los cambios que se han aplicado a los datos. Este archivo es requerido para la recuperación del sistema

Como mínimo, cada base de datos contendrá un archivo de datos y un archivo de registro de transacciones y no hay dos bases de datos que puedan compartir los mismos datos o archivo de registro.

2.7. Oracle

Las bases de datos Oracle pueden clasificarse en función de su funcionalidad:

- Procesado transaccional en línea (Online Transactional Processing – OLTP): Orientada a un gran número de transacciones en poco tiempo. La respuesta es rápida y con integridad de los datos a la hora de realizar gran cantidad de inserciones, actualizaciones o borrados de registros. Suele ser el modelo de datos más usual
- Procesado analítico en línea (Online Analytic Processing – OLAP): Orientado a análisis de datos. El volumen de transacciones es pequeña, pero de gran volumen de datos. Este tipo de base de datos está orientada a la respuesta rápida de datos calculados.

2.7.1. Elementos de una base de datos Oracle

Una base de datos Oracle está compuesta de tres partes:

Memoria

La memoria puede ser asociada a los procesos de usuario (PGA – Process Global Area) o asociada al servidor Oracle (SGA – System Global Area)

La SGA son buffers que Oracle toma del sistema operativo, cuando arranca la base de datos. La porción de memoria, que toma inicialmente y como lo distribuye en sus diferentes componentes, está determinado por el fichero de inicialización de la base de datos. En la SGA se sitúan datos, información del diccionario de datos, gestión de bloqueos, etc.…

Procesos

Existen dos tipos de procesos:

- Procesos de usuario, sqlplus, sqlforms….
- Procesos background o procesos de Oracle como DBWR, LGWR…

Ficheros de datos

Existen varios tipos de ficheros:

- Ficheros de parámetros: spfile/pfile, listener, tnsnames,…
- Archivo de control (control file): Contienen información para poder arrancar la base de datos, su nombre, fecha de creación, ficheros asociados a la base de datos,…

- Archivo de datos (datafile): Son los ficheros donde se almacena la información de la base de datos.
- Archivo de rehacer (redo log file): Sirven para mantener la consistencia en caso de fallo

2.7.2. Instancias

Cuando Oracle se inicia se le asigna una porción de memoria para su funcionamiento. Esta área de memoria, conocido como Area Global de Sistema (SGA), se divide en diferentes estructuras. Junto con el espacio de memoria también se inician una serie de procesos que interactúan con el SGA. El espacio de memoria y los procesos constituyen una instancia de Oracle. Una instancia de Oracle podría funcionar si su base de datos accesible.

Cuando se inicia un sistema de Oracle, la instancia se crea en la memoria, a continuación se conecta a la base de datos y finalmente esta se abre para la iteración del usuario. Cuando el sistema se apaga, la instancia se borra de la memoria y los procesos se terminan, pero la base de datos continúa en el disco aunque en un estado cerrado.

Una instancia tendrá una base de datos asociada a ella y una base de datos podrá tener una o más instancias para acceder a ella.

En Oracle, la agrupación lógica de los objetos de la base de datos se realiza a través de tablespaces, en ella se agrupan las tablas, vistas, índices y otros objetos. Cada tablespace está físicamente representado por uno o más archivos de datos en disco y forma parte de la base de datos.

2.8. Seguridad en Bases de Datos

La mayor parte de la información de las organizaciones se almacena en bases de datos y es fundamental proteger esta información de ataques maliciosos, sean estos internos o externos.

Al estar la mayor parte de la información sensible almacenada en los en sistemas gestores de bases de datos, provoca que los hackers centren gran parte de su esfuerzo en acceder a esa información por medio de alguna de las muchas vulnerabilidades que se encuentran en esos gestores. Dichas vulnerabilidades pueden ser debidas a problemas de seguridad en el software, por cómo está configurado su acceso o por problemas en la programación de la aplicación que accede a la información.

2.8.1. Tipos de ataques

En líneas generales, se pueden clasificar los ataques en dos tipos:

- Ataques que no requieren autenticación: Son los ataques más comunes porque no se necesita usuario y contraseña. Suelen ser ataques externos y aprovechan alguna vulnerabilidad en la base de datos o en las aplicaciones que acceden a ella.
- Ataques que requieren autenticación: Este tipo de ataques se lanzan por personas que tienen las claves de acceso, bien porque se han conseguido de forma ilícita o bien porque son usuarios internos y autorizados.

2.8.2. Vulnerabilidades más comunes en bases de datos

Según el WhitePaper de Acens Technologies [18 Bases de datos y sus vulnerabilidades más comunes], perteneciente a Telefónica, entre las vulnerabilidades más comunes que podemos encontrar en las bases de datos están las siguientes:

Usuario o contraseña en blanco o una contraseña débil.

El usuario y contraseña son la primera defensa de entrada a la información de la base de datos y se debe optar por utilizar un acceso más complejo.

Otorgar privilegios innecesarios a usuarios.

En muchas ocasiones los usuarios disponen de más privilegios de los que necesitan sobre la base de datos. Es recomendable limitar los privilegios a lo estrictamente necesario y autorizado.

Características innecesarias habilitadas.

Generalmente los gestores de bases de datos vienen con características y módulos a instalarse por defecto y no siempre son necesarios. Eso abre más puertas de entrada y más riesgos si alguno de esos paquetes tiene alguna vulnerabilidad.

Desbordamiento de búfer.

Se desborda la aplicación por el exceso de información y peticiones que recibe.

Falta de actualizaciones.

El no tener las últimas actualizaciones instaladas implica un riesgo ya que estas solucionan problemas de seguridad detectados.

Datos sensibles sin cifrar.

No siempre se cifra la información sensible. Cifrándola se consigue complicar la compresión de la información si se ha tenido acceso a ella por parte de algún atacante.

Inyecciones SQL.

Una mala limpieza de los datos de entrada puede provocar que se ejecute código o sentencias no deseadas pudiendo dar acceso sin restricciones para copiar o modificar datos.

Denegación de servicio (DoS)

A través de este ataque se niega el acceso a la información a usuarios autorizados.

Exposición de los medios de almacenamiento para backup.

A menudo se tiene protegida la base de datos, pero no se emplea la misma seguridad para proteger las copias de seguridad.

Ataques de fuerza bruta o diccionario

Consiste en probar todas las combinaciones posibles para intentar reventar una entrada de la aplicación.

Un estudio realizado por la empresa Imperva en el 2015 [19 Top Ten Database Security Threats] señala el top ten de las amenazas que afectan a las bases de datos.

En ese estudio se puede observar que las amenazas no han variado desde el último estudio realizado en el 2013 y también se detallan las posibles soluciones divididas en seis categorías diferentes:

- Descubrimiento y Evaluación. Localizar donde residen las vulnerabilidades y los datos críticos.
- Gestión de derechos de usuario. Identifica derechos excesivos sobre datos sensibles.
- Seguimiento y bloqueo. Proteger las bases de datos de ataques, acceso no autorizado y robo de los datos.
- Auditoría. Ayuda a demostrar el cumplimiento de las regulaciones.
- Protección de Datos. Garantiza la integridad y la confidencialidad de los datos.
- Seguridad no técnica. Inculca y refuerza una cultura de concienciación sobre la seguridad y la formación.

2.8.3. Errores humanos

Si en las secciones anteriores hemos visto los tipos de ataque y vulnerabilidades más comunes, en esta parte vamos a explorar los tipos de errores más comunes y sus impactos sobre las bases de datos.

1. Errores humanos del usuario final

Los usuarios pueden cometer errores que podrían tener un impacto serio. Sin embargo, y debido a que no suelen tener grandes privilegios, los errores suelen ser limitados y de naturaleza localizada

2. Uso inapropiado de las aplicaciones

Al usarse las aplicaciones inapropiadamente puede ocurrir lo siguiente:

- Archivos sobrescritos inadvertidamente.
- Datos incorrectos utilizados como entrada de una aplicación.
- Archivos no claramente nombrados u organizados.
- Archivos borrados accidentalmente.
- Etc.

3. Errores del administrador de sistemas

A diferencia del usuario final, los administradores del sistema tienen todo tipo de privilegios y acceso. En consecuencia, sus errores tienen mayor impacto y son más difíciles de restablecer.

2.9. Seguridad en SGBD

2.9.1. Seguridad en MySQL

Al hablar de seguridad hay que tener en cuenta que no solo hay que proteger el servidor MySQL, sino toda la maquina del servidor (Sistema operativo, acceso al hardware, etc.)

MySQL utiliza seguridad basada en Access Control Lists (ACL) para todas las conexiones, consultas y otras operaciones que los usuarios pueden intentar realizar. También tiene soporte para conexiones cifradas con SSL.

Para proteger el servidor MySQL se deben seguir una serie de pautas en medidas de seguridad que se explican en detalle en su página oficial. [16 MySQL Security]

- El acceso a la tabla de usuario de MySQL solo puede ser realizada por el usuario root, por ningún usuario más.
- Se debe utilizar GRANT y REVOKE para controlar el acceso a MySQL. No se deben otorgar más privilegios de los necesarios. Nunca otorgar privilegios a todos los hosts.
- Nunca guardar las contraseñas en texto plano. Utilizar SHA2, SHA1, MD5 o alguna otra función para almacenar el valor hash.
- Todas las cuentas deben tener una contraseña y no se deben de utilizar contraseñas de diccionarios.
- Utilizar un Firewall para proteger el servidor. El puerto que utilice MySQL no debe ser accesible desde hosts no confiables.
- Las aplicaciones que accedan a MySQL no deben confiar en los datos introducidos por los usuarios, y deben ser escritas utilizando técnicas de programación defensivas.
- No se deben transmitir datos sin cifrar a través de Internet. Se deben utilizar protocolos de cifrado como SSL o SSH. MySQL soporta conexiones SSL internas.
- Cuando se conecta a un servidor MySQL se debería utilizar una contraseña. A partir de la versión 4.1.1 las contraseñas no se transmiten en texto claro. El resto de información se transfiere en forma de texto y si la conexión pasa por una red insegura se podría utilizar el protocolo comprimido para que sea más difícil de descifrar. También se podrían utilizar conexiones SSL.
- La única cuenta del sistema operativo que debe tener privilegios de lectura o escritura en los directorios de la base de datos debe ser la cuenta que se utiliza para ejecutar mysqld.

- El servidor MySQL no debe funcionar con el mismo usuario que el administrador del sistema operativo.
- No otorgar el privilegio FILE para usuarios no administrativos. Cualquier usuario que tenga este privilegio podrá escribir un archivo en cualquier parte del sistema de archivos con los privilegios del demonio mysqld.
- No otorgar el privilegio PROCESS o SUPER para usuarios no administrativos. El privilegio SUPER puede utilizarse para cerrar conexiones de cliente.
- No permitir el uso de enlaces simbólicos a tablas. Esto es especialmente importante si se ejecuta mysqld como administrador, ya que cualquiera que tenga acceso de escritura al directorio de datos del servidor entonces podrá borrar cualquier archivo del sistema.
- Si no se confía en el servidor DNS, se deben utilizar direcciones IP en lugar de nombres de hosts en las tablas de permisos.
- Si se desea restringir el número de conexiones permitidas a una sola cuenta, se puede hacer estableciendo la variable max_user_connections en mysqld

2.9.1.1. Autentificación

Al ser MySQL un sistema multiusuario es necesario establecer la autentificación y administración de usuarios, la administración de privilegios y contraseñas y establecer límites de recursos de la base de datos.

Cualquier usuario que intente conectarse a MySQL deberá realizarlo con un nombre de usuario para que el servidor autentifique que dicha persona está autorizada a usar la cuenta y el inicio de sesión.

Existen cisco tablas que controlan la validación de permisos en MySQL:

- User. Contiene los usuarios que pueden conectarse con el servidor con cualquier privilegio que tengan.
- Db. Contiene los privilegios de nivel de base de datos.
- Tables-priv. Contiene los privilegios de nivel de tabla.
- Columns_priv. Contiene los privilegios de nivel de columna.
- Host. Se usa en combinación con la tabla db para controlar los privilegios de acceso de hosts particulares al mejor nivel que sea posible solo en la tabla db

2.9.2. Seguridad en PostgreSQL

Tal y como se indica en la página oficial de PostgreSQL [22 PostgreSQL Documentation], la seguridad de la base de datos esta implementada en varios niveles:

- Protección de los ficheros de la base de datos. Todos los ficheros almacenados en la base de datos están protegidos contra escritura por cualquier cuenta distinta al súper usuario.
- Por defecto, las conexiones de los clientes al servidor de la base de datos están permitidas, únicamente mediante sockets locales y no mediante sockets TCP/IP. Se pueden permitir las conexiones no locales arrancando indicando esa opción.
- Las conexiones de los clientes se pueden restringir por dirección IP y/o por nombre de usuario.
- Las conexiones de los clientes pueden ser autentificadas mediante otros paquetes externos.
- Por defecto, los usuarios no tienen permiso de escritura a bases de datos que no hayan creado.
- Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse en base a esos grupos.

2.9.2.1. Autentificación

Mediante la autentificación nos aseguramos de que el usuario que está solicitando acceso a la base de datos es en realidad quien dice ser. La comprobación de los usuarios se realiza en la tabla *pg_user*.

La verificación de la identidad del usuario se realiza de dos formas distintas:

1. Desde la shell del usuario.
2. Desde la red

2.9.2.2. Copia de seguridad y restauración

Al gestionar PostgreSQL sus propios ficheros en el sistema, no es conveniente confiar en terceras aplicaciones de backup para realizar las copias de seguridad.

Postgres proporciona dos utilidades para realizar las copias de seguridad de su sistema:

- pg_dump: Copias de seguridad de bases de datos individuales.
- pg_dumpall: Copias de seguridad de toda la instalación de una sola vez.

2.9.3. Seguridad en Microsoft SQL Server

SQL Server proporciona una arquitectura de seguridad que permite a los administradores contrarrestar las amenazas. Cada versión de SQL Server ha ido mejorando en características de seguridad y funcionalidades. [23 Información general sobre seguridad de SQL Server]

Cada instancia de SQL Server tiene un conjunto jerárquico de entidades, empezando por el servidor. Cada servidor tiene varias bases de datos y cada base de datos una serie de objetos susceptibles de ser protegidos.

Cada SQL Server a proteger tiene permisos asociados que se pueden conceder a una entidad de seguridad, la cual puede ser un elemento único, un grupo o un proceso al que se le otorga permiso de acceso a SQL Server.

La arquitectura de seguridad administra el acceso a entidades protegidas mediante autenticación y autorización.

- Autenticación: Es el proceso de inicio de sesión por el que una entidad solicita el acceso mediante el envío de credenciales que el servidor SQL Server evalúa. La autenticación establece la identidad del usuario o del proceso que se autentica.
- Autorización: Es el proceso con el que se determinan los recursos a proteger a los que tiene acceso una entidad, así como las operaciones que les están permitidas.

2.9.3.1. Autentificación

SQL server admite dos modos de autenticación, el modo mixto y el modo de autenticación de Windows.

El modo predeterminado de autenticación es la autenticación de Windows. Este modo de seguridad de SQL Server está integrado con la seguridad de Windows. Para iniciar la sesión en SQL Server se confía en las cuentas de usuario de Windows, y si estos ya han sido autenticados en Windows, no tienen que presentar credenciales adicionales.

El modo mixto admite la autenticación de Windows y la autenticación de SQL Server. Los pares de nombre de usuario y contraseña se mantienen en SQL Server.

La autenticación de Windows es más segura porque utiliza una serie de mensajes cifrados para autenticar los usuarios en SQL server. Cuando se utiliza la autenticación de SQL

Server, los nombres y contraseñas viajan a través de la red, lo que convierte este método en menos seguro.

Generalmente la autenticación de Windows es la mejor opción en las situaciones siguientes:

- Cuando existe un controlador de dominio.
- Cuando la aplicación y la base de datos se encuentran en el mismo equipo.
- Se está utilizando una instancia de SQL Server Express o LocalDB.

Y la autenticación en SQL Server en las siguientes:

- Cuando existe un grupo de trabajo.
- Cuando las conexiones se realizan desde distintos dominios que no son de confianza.
- Cuando se utilizan aplicaciones de Internet.

Tipos de Inicio de sesión

SQL Server admite los siguientes tipos de inicio de sesión:

- Una cuenta de usuario de Windows local o de un dominio de confianza. En este caso, SQL Server se basa en Windows para autenticar las cuentas.
- Grupo de Windows. Cuando se concede acceso a SQL Server a un grupo de Windows, el acceso es concedido a todos los usuarios miembros de dicho grupo.
- Inicio de sesión de SQL Server. Se almacena el nombre de usuario y un valor hash de la contraseña en la base de datos maestra.

2.9.3.2. Autorización y permisos en SQL Server

Cuando se crean objetos de bases de datos, se deben de conceder los permisos de forma explícita para que los usuarios tengan acceso a ellos. Cada objeto a proteger tiene permisos que se pueden conceder a una entidad mediante instrucciones de permiso.

El principio de privilegios mínimos (LUA) garantiza que los usuarios inician sesión con cuentas de usuario limitadas. Es importante seguir este principio cuando se conceden permisos a usuarios de base de datos, otorgando a usuarios y roles los mínimos permisos necesarios para realizar una tarea concreta.

Los permisos se otorgan a través de las instrucciones GRANT, REVOKE y DENY

- GRANT: Concede un permiso.
- REVOKE: Revoca un permiso. Por defecto, este es el estado de un objeto nuevo. Un permiso revocado a un usuario (o rol) se puede heredar de otros grupos (o roles) a los que esta asignada la entidad de seguridad.
- DENY: Revoca un permiso de manera que no pueda ser heredado. DENY tiene prioridad sobre todos los permisos pero no se aplica a los propietarios del objeto.

Permisos basados en roles

El otorgar permisos a los roles en vez de a los usuarios simplifica la administración de seguridad. Estos permisos otorgados a los roles se heredan por parte de todos los miembros pertenecientes a ese rol.

Permisos a procedimientos o funciones

Encapsular el acceso a los datos a través de procedimientos almacenados o funciones brinda un nivel de protección adicional. De esta manera se evita que los usuarios interactúen directamente con los objetos de la base de datos otorgando los permisos únicamente a procedimientos almacenados o funciones y denegando permisos a objetos tales como tablas.

2.9.3.3. Cifrado de datos

SQL Server ofrece la opción de cifrar y descifrar datos mediante un certificado, una clave asimétrica o una clave simétrica. Estas opciones se administran en un almacén de certificados interno que utiliza una jerarquía de cifrado que protege los certificados y las claves.

El método más rápido de cifrado es el de clave simétrica, este modo es el más adecuado a la hora de controlar grandes volúmenes de datos. Las claves simétricas se pueden cifrar mediante certificados, contraseñas u otras claves simétricas.

SQL Server admite distintos algoritmos de cifrado de clave simétrica, tales como DES, Triple DES, RC2, RC4, RC4 de 128 bits, DESX, AES de 128 bits, AES de 192 bits y AES de 256 bits.

2.9.4. Seguridad en Oracle

Oracle proporciona mecanismos de seguridad que controlan el acceso y el uso de los datos. Entre otras cuestiones, estos mecanismos se ocupan de lo siguiente:

- Prevenir el acceso no autorizado a las bases de datos y objetos.
- Controlar el uso del disco y de los recursos del sistema.
- Monitorizar las acciones de los usuarios.

Cada usuario de Oracle tiene asociado un esquema con el mismo nombre. Los esquemas son un conjunto lógico de objetos (tablas, vistas, índices, funciones, procedimientos, etc.) y cada usuario tiene acceso a todos los objetos de su correspondiente esquema.

La seguridad en la base de datos Oracle se puede clasificar en dos categorías: Seguridad del sistema y seguridad de los datos.

- Seguridad del sistema. Proporciona los mecanismos de seguridad para controlar el acceso y el uso de la base de datos a nivel del sistema. Entre los mecanismos que proporciona están:
    - Validación de usuario y contraseña.
    - Control de la cantidad de espacio en disco disponible para los objetos de los usuarios.
    - Limitación de los recursos de cada usuario.

- Seguridad de los datos. Proporciona los mecanismos de seguridad para controlar el acceso y uso de la base de datos a nivel de objetos. Entre los mecanismos que proporciona están:
    - Control de los usuarios que acceden a los esquemas y las acciones que pueden ejecutar.
    - Monitorización de cada esquema.

Oracle proporciona un control de acceso discrecional, el cual es un medio de restricción de acceso basado en privilegios. Para que cualquier usuario pueda acceder a un objeto, se le deben otorgar previamente los privilegios apropiados. Los usuarios con privilegios apropiados podrán otorgar privilegios a otros usuarios.

2.9.4.1. Mecanismos de Seguridad

La administración de la base de datos de Oracle se realiza utilizando diferentes servicios:

Usuarios y esquemas

Cada base de datos de Oracle tiene asociada una lista de nombres de usuarios. Para acceder a una base de datos, un usuario deberá autentificarse ante la misma.

Cada usuario tiene un dominio de seguridad que definen un conjunto de propiedades que determinan factores como:

- Acciones (privilegios y roles) disponibles para cada usuario.
- Limites de espacio en el Tablespace para cada usuario.
- Limites de utilización de recursos del sistema para cada usuario.

Privilegios

Los privilegios son permisos para ejecutar un tipo particular de sentencias SQL. Estos pueden ser divididos en dos categorías, privilegios del sistema y privilegios de objetos.

- Privilegios de sistema. Permiten a los usuarios desempeñar una acción particular dentro del sistema, como puede ser crear un tablespace. Muchos de estos privilegios están disponibles únicamente para los administradores del sistema.
- Privilegios de objetos. Permiten desempeñar acciones sobre un esquema específico. Estos privilegios son otorgados a los usuarios finales.

Los privilegios pueden ser asignados a los usuarios explícitamente o a roles, y estos a su vez ser asignados a uno o más usuarios.

Roles

Los roles ofrecen una administración más sencilla y controlada de los privilegios. Estos son asignados a usuarios o a otros roles.

Los privilegios se asignan a los roles para que los hereden sus usuarios. Cuando los privilegios de un grupo de usuarios deben cambiar, únicamente será necesario modificar los privilegios del rol al que pertenecen.

Configuración del almacenamiento y quotas

Oracle proporciona los recursos para limitar el uso del espacio en disco para cada usuario. Además también se puede limitar el espacio disponible de disco colectivo. Estas quotas o límites de espacio pueden ser configurados para cada tablespace, permitiendo un control selectivo sobre el espacio en disco que es ocupado por cada objeto de cada esquema.

Limites de los recursos

Cada usuario de Oracle tiene asignado un perfil que especifica las limitaciones sobre los recursos del sistema. Entre estas limitaciones están:

- Numero de sesiones concurrentes que se pueden establecer.
- Tiempo de CPU disponible para la sesión de usuario o para una llamada a Oracle realizada por una sentencia SQL.
- Cantidad de tiempo ocioso o de conexión para cada sesión de usuario.
- Restricción en el tipo de contraseñas y en su reutilización, así como el bloqueo de cuentas tras una determinada cantidad de intentos de conexión fallidos.

2.10. Auditoría en Bases de Datos

La auditoría permite monitorizar y registrar los accesos a la información almacenada en las bases de datos. Permite determinar:

- Quien accede a los datos.
- Cuando se accede a los datos.
- Desde donde y como se accede a los datos.
- La sentencia SQL utilizada para acceder a los datos.
- El efecto causado por el acceso a los datos.

Mediante la auditoría de las bases de datos se puede:

- Monitorizar y registrar el uso de los datos por los usuarios, ya estén estos autorizados o no.
- Mantener trazas de uso y acceso a los datos.
- Generar alertas en tiempo real.

2.10.1. Auditoría en MySQL

MySQL, a diferencia de otros gestores de bases de datos como Oracle o MS SQL server, no dispone de un servicio de auditoría como tal. Se pueden utilizar disparadores o triggers con ese fin, de tal manera que se guarden en una tabla auxiliar todas las modificaciones que se vayan realizando en todos los campos de una tabla de la base de datos.

A través de los trigger podremos saber que modificaciones se han realizado, con que usuario, en qué fecha y hora y que valores había antes y después de la modificación.

A diferencia de la auditoría, mediante los triggers únicamente se registran las transacciones en las que existe algún tipo de modificación sobre los datos, pero no cuando únicamente se produce un acceso a ellos sin alteración de los mismos.

En el punto “3.3.1. MySQL. Auditoría” de esta memoria se especifica cómo se crean los Triggers en MySQL.

2.10.2. Auditoría en PostgreSQL

De igual manera que MySQL y a diferencia de otros gestores de bases de datos como Oracle o MS SQL server, PostgreSQL no dispone de un servicio de auditoría como tal. Para registrar el uso de los datos por parte de los usuarios o aplicaciones se pueden utilizar disparadores o triggers, de esta manera se guardan en una tabla auxiliar todas las modificaciones que se vayan realizando en todos los campos de una tabla de la base de datos.

A través de citados trigger se pueden registrar las modificaciones que se han realizado sobre los datos, con que usuario, en qué fecha y hora y que valores había antes y después de la modificación.

Como ya se ha comentado en el caso de MySQL, mediante los triggers únicamente se registran las transacciones en las que existe algún tipo de modificación sobre los datos, pero no cuando únicamente se produce un acceso a ellos sin alteración de los mismos.

En el punto “3.3.2. PostgreSQL. Auditoría” de esta memoria se especifica cómo se crean los Triggers en PostgreSQL.

2.10.3. Auditoría en Microsoft SQL Server

La auditoría de una base de datos o de una instancia de SQL Server registra todos los eventos que se producen en el sistema. La auditoría se realiza a nivel de instancia y es posible tener varias auditorías por cada instancia.

Hay varios niveles de auditoría disponibles en SQL Server, dependiendo de los requisitos de cada instalación. Todas las ediciones de SQL Server admiten auditorías a nivel de servidor, las auditorías a nivel de base de datos se limitan a las ediciones Enterprise, Developer y Evaluation.

El registró y visualización de los cambios realizados en las distintas tablas de SQL Server se puede llevar a cabo a través de diferentes medios:

- Auditoría
- Triggers
- Change Data Capture (CDC)
- SQL Server Profiler

En el punto “3.3.3. MS SQL Server. Auditoría” de esta memoria se especifica cómo se crean los distintos tipos de auditoría en Microsoft SQL Server.

En el punto “3.3.3.1. Mediante Auditoría” se crea la auditoría de MS SQL Server.

En el punto “3.3.3.2. Mediante Triggers” se crean los triggers en MS SQL Server.

En el punto “3.3.3.3. Mediante SQL Server Profiler” se activa y configura SQL Server profiler.

2.10.4. Auditoría en Oracle

Existen dos tipos de auditoría principal en Oracle, la auditoría estándar (Standard Auditing) y la auditoría de grano fino FGA (Fine Grained Auditing), que extiende la auditoría estándar y, además, captura la sentencia SQL que ha sido ejecutada.

Standard auditing nos permite tener un registro de información acerca de una operación hecha en base de datos (información como usuario de ejecución, fecha, terminal). Esta opción de auditoría está incluida dentro de la licencia Standard Edition y Enterprise Edition.

FGA nos permite registrar operaciones en base a políticas de auditoría, como acciones sobre campos específicos, horarios de ejecución, y condiciones de una ejecución. Esta opción de auditoría está incluida dentro de la licencia Enterprise Edition.

Además de las auditorías citadas, se pueden crear triggers sobre las tablas para llevar el registro de los cambios realizados sobre ellas.

En el punto “3.3.4. Oracle. Auditoría” de esta memoria se especifica cómo se crean los distintos tipos de auditoría en Oracle.

En el punto “3.3.4.1. Auditoría Estándar” se crea la auditoría estándar de Oracle.

En el punto “3.3.4.2. Fine Grained Auditing (FGA)” se crea la auditoría de grano fino de Oracle.

3. Análisis forense

En el análisis forense sobre sistemas gestores de bases de datos, vamos a trabajar con cuatro escenarios diferentes instalados en un servidor Windows 2012:

1. MySQL, versión 5.6.17
2. PostgreSQL, versión 9.4
3. Microsoft SQL Server, versión 2014
4. Oracle, versión 12c

Se realizarán cuatro tipos de ataque distintos y posteriormente se intentará obtener la mayor información posible de los ataques realizados **procurando no utilizar para ello ningún software de terceros**, para obtener la información intentaremos utilizar únicamente las utilidades del propio gestor de base de datos si es posible.

Como ya se ha comentado anteriormente, en líneas generales, se pueden clasificar los ataques en dos tipos:

- Ataques que no requieren autenticación: Son los ataques más comunes porque no se necesita usuario y contraseña. Suelen ser ataques externos y aprovechan alguna vulnerabilidad en la base de datos o en las aplicaciones que acceden a ella.
- Ataques que requieren autenticación: Este tipo de ataques se lanzan por personas que tienen las claves de acceso, bien porque se han conseguido de forma ilícita o bien porque son usuarios internos y autorizados.

En la elección de los cuatro ataques a realizar, se han seleccionado dos que requieren autenticación y otros dos que no la requieren.

Los dos ataques que requieren autenticación se realizaránuna por parte del administrador de la base de datos y otra por un usuario autorizado, bien de forma intencionada o por error.

Entre los ataques que no requieren autenticación, hemos intentado seleccionar los más característicos y habituales, por un lado los que se aprovechan de alguna vulnerabilidad de la aplicación que accede a la base de datos (SQL Injection) y por otro un ataque directo a la base de datos (fuerza bruta).

La vulnerabilidad SQL Injection se basa en la no validación de los datos de entrada en la aplicación que accede a la base de datos pudiéndose ejecutar código o sentencias no

deseadas. El ataque de fuerza bruta consiste en probar todas las combinaciones posibles para intentar reventar el acceso a la base de datos.

En resumen, los ataques a realizar son:

- Ataque interno: Usuario administrador que modifica datos.
- Ataque interno: Usuario de la empresa que modifica datos.
- Ataque externo: SQL Injection.
- Ataque externo: Fuerza bruta.

Los ataques se realizaránteniendo el gestor de bases de datos de dos modos diferentes:

- Únicamente registrando logs.
- Con auditoría.

3.1. Entorno experimental

El entorno experimental sobre el que se realiza el trabajo se basa en cuatro escenarios virtualizados con VirtualBox con sistema operativo Microsoft Windows 2012 Server sobre los que se implementan los gestores de bases de datos. Los ataques se realizan desde dos escenarios virtualizados, el primero con sistema operativo Windows 7 profesional y el segundo con Linux Kali

- Primer escenario:

Este primer escenario consta de Microsoft Windows 2012 Server con el sistema gestor de bases de datos MySQL, versión 5.6.17, instalado mediante WAMP Server.

- Segundo escenario:

El segundo escenario también con Microsoft Windows 2012 Server tiene el SGBD PostgreSQL, versión 9.4.

- Tercer escenario:

Microsoft Windows 2012 Server con Microsoft SQL Server, versión 2014, como sistema gestor de bases de datos.

- Cuarto escenario:

El cuarto escenario contiene Oracle, versión 12c y Microsoft Windows 2012 Server como sistema operativo.

En cuanto a los dos escenarios desde los que se realizan los ataques son:

- Quinto escenario:

Windows 7 profesional con Access 2007 para realizar el segundo y tercer ataque.

- Sexto escenario:

Sistema operativo Linux Kali con las herramientas nmap e Hydra para realizar el cuarto ataque.

3.2. Creación de la base de datos

Para los ataques y la posterior obtención de información vamos a suponer que tenemos una base de datos con productos para la venta.

Esta base de datos llamada Unir, contendrá una tabla llamada “Productos” en la que se introducirán los registros de tres productos con su nombre, precio y stock.

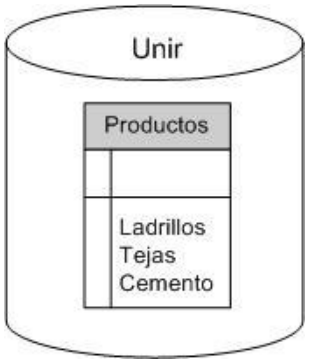


Figura 5. Base de Datos

- **Base de datos:**
  - MySQL

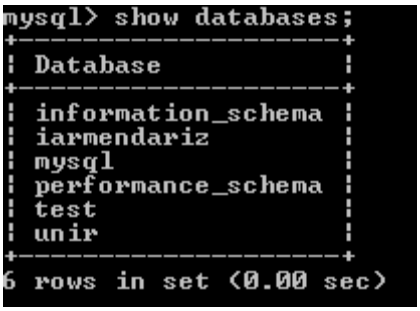


Figura 6. Base de datos MySQL

  - PostgreSQL

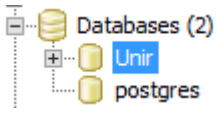


Figura 7. Base de datos PostgreSQL

- Microsoft SQL Server

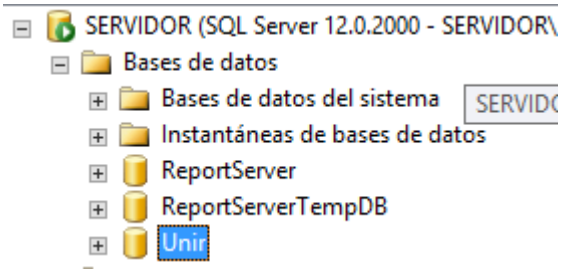


Figura 8. Base de datos MS SQL Server

- Oracle

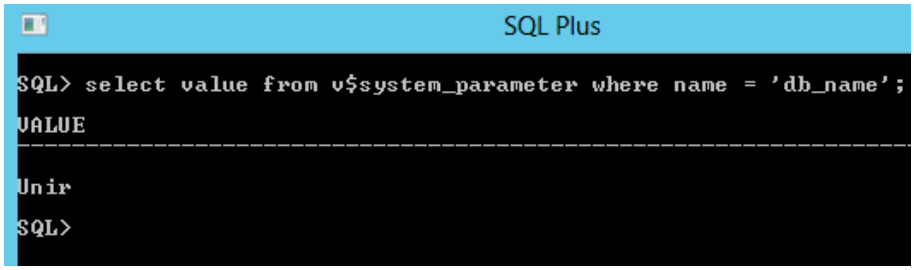


Figura 9. Base de datos Oracle

- **Tabla Productos:**
  - MySQL

```
mysql> select * from productos;
+------------+-----------+--------+-------+
| IdProducto | Producto  | Precio | Stock |
+------------+-----------+--------+-------+
|          0 | Ladrillos |     13 |    20 |
|          1 | Tejas     |     20 |    40 |
|          2 | Cemento   |     50 |   100 |
+------------+-----------+--------+-------+
3 rows in set (0.00 sec)
```

Figura 10. Tabla Productos MySQL

  - PostgreSQL

	idproducto [PK] serial	producto character varying(30)	precio numeric	stock numeric
1	0	Ladrillos	13	20
2	1	Tejas	20	40
3	2	Cemento	50	100
*				

Figura 11. Tabla Productos PostgreSQL

- Microsoft SQL Server

SQLQuery1.sql - SE...R\iarmendariz (56))* | SERVIDOR.Unir - dbo.productos

	IdProducto	Producto	Precio	Stock
	0	Ladrillos	13	20
	1	Tejas	20	40
	2	Cemento	50	100
▶*	NULL	NULL	NULL	NULL

Figura 12. Tabla Productos MS SQL Server

- Oracle

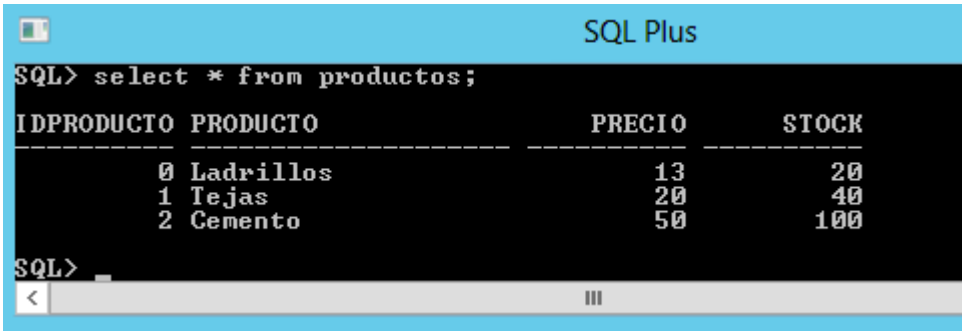


Figura 13. Tabla Productos Oracle

3.3. Registros (Logs)

3.3.1. MySQL. Logs

MySQL genera varios logs en los que se registra su actividad:

Tipo de registro	Información
Error log	Errores al iniciar, correr o detener el motor de MySQL
General query log	Conexiones establecidas y declaraciones recibidas de los clientes (tanto si modifican datos como si no)
Binary log	Declaraciones que modifican datos (se utiliza también para la replicación)
Relay log	Cambios en los datos recibidos de un servidor maestro de replicación
Slow query log	Registro de las consultas que necesitan más tiempo en ejecutarse que el establecido en la variable long_query_time
DDL log (metadata log)	Operaciones de metadatos realizadas por sentencias DDL

Tabla 5. Logs de actividad MySQL

El log de consultas o GENERAL LOG de MySQL, es uno de los ficheros de registro de log que registra las conexiones de clientes establecidas, y las sentencias ejecutadas.

Para activarlo, si no lo está ya, habrá que modificar el fichero de configuración de MySQL indicando donde se quiere registrar el log

```
general-log=1
general-log-file = "C:\wamp\logs\mysql_general.log"
```

Figura 14. General Log MySQL

3.3.2. PostgreSQL. Logs

La activación y desactivación de los registros de log de PostgreSQL se realiza en el fichero de configuración postgresql.conf

```
log_connections = on
log_disconnections = on
log_duration = on
#log_error_verbosity = default          # terse, default, or verbose messages
log_hostname = on
log_line_prefix = '%t '                 # special values:
                                        #   %a = application name
                                        #   %u = user name
                                        #   %d = database name
                                        #   %r = remote host and port
                                        #   %h = remote host
                                        #   %p = process ID
                                        #   %t = timestamp without milliseconds
                                        #   %m = timestamp with milliseconds
                                        #   %i = command tag
                                        #   %e = SQL state
                                        #   %c = session ID
                                        #   %l = session line number
                                        #   %s = session start timestamp
                                        #   %v = virtual transaction ID
                                        #   %x = transaction ID (0 if none)
                                        #   %q = stop here in non-session
                                        #        processes
                                        #   %% = '%'
                                        # e.g. '<%u%%%d> '
#log_lock_waits = off                   # log lock waits >= deadlock_timeout
log_statement = 'all'                   # none, ddl, mod, all
```

Figura 15. Logs PostgreSQL

3.3.3. MS SQL Server. Registro de Transacciones

Todas las bases de datos de SQL Server tienen un registro de transacciones que registra todas las transacciones y las modificaciones que cada transacción realiza en la base de datos.

Este registro es un componente esencial de la base de datos y, si se produce un error del sistema, podría ser necesario para volver a poner la base de datos en un estado coherente. El registro de transacciones nunca se debe eliminar o mover, a menos que se conozcan las implicaciones de esas acciones.

El registro de transacciones permite las siguientes operaciones:

- Recuperación de transacciones individuales.
- Recuperación de todas las transacciones incompletas cuando se inicia SQL Server.

- Puesta al día de una base de datos, un archivo, un grupo de archivos o una página restaurados hasta el momento exacto del error.
- Permitir replicación transaccional.
- Permitir soluciones de servidor en espera.

El archivo con este registro se guarda en formato LDF, pero si no se utiliza alguna herramienta especial para visualizarlo muestra entradas ininteligibles, así que estas entradas no pueden ser leídas directamente.

3.3.4. Oracle. Archivos redo log

Los ficheros de redo log registran los cambios realizados en la base de datos como resultado de transacciones o acciones internas del servidor Oracle.

Cuando el modo archive log esta activado en una base de datos Oracle, protege contra la pérdida de datos si se produce un fallo en el medio físico.
- Se puede realizar una copia de seguridad mientras la base de datos está on-line.
- Con este modo de base de datos se puede restaurar una copia de seguridad de los archivos dañados utilizando estos archivos para actualizar los archivos mientras están online.
- Se puede recuperar la base de datos en un número de cambio del sistema específico.
- Se puede restaurar la base de datos en un punto específico en el tiempo.

Para saber en qué modo esta la base de datos podemos utilizar el comando “archive log list”

Figura 16. Archive log list Oracle

Una vez activado, el resultado será el siguiente:

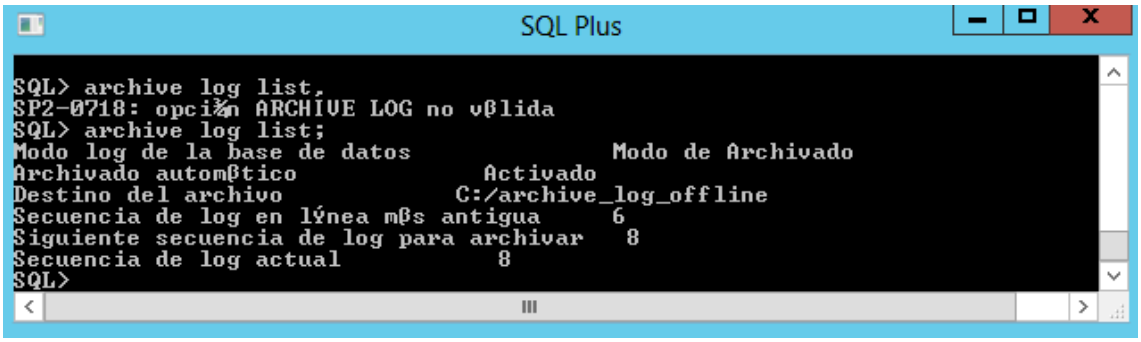


Figura 17. Archive log list Oracle

Para poder interpretar estos registros de redo log, Oracle proporciona la utilidad llamada LogMiner, básicamente lo que hace es leer o recorrer los redo logs, viendo su contenido para efectos de búsqueda de transacciones, información, etc.

A la hora de utilizar LogMiner tenemos que tener en cuenta dos conceptos, el diccionario LogMiner y los archivos de registro (Redo logs) que contienen los datos que queremos visualizar o recuperar.

El diccionario de LogMiner proporciona los nombres de las tablas y columnas, en lugar de los ID internos de los objetos. LogMiner utiliza el diccionario para traducir los identificadores internos de objetos a los nombres de los objetos que puedan ser interpretados. Sin un diccionario, los datos que se mostraránserán los ID de los objetos y presentará los datos en modo binario.

Para crear el diccionario que utiliza LogMiner habrá que usar el siguiente comando:

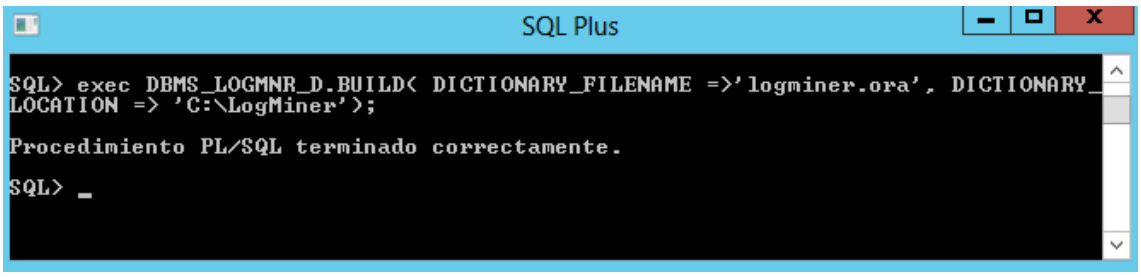


Figura 18. Diccionario LogMiner Oracle

Ahora solamente tenemos que indicarle a logminer los ficheros redo que necesitamos cargar:

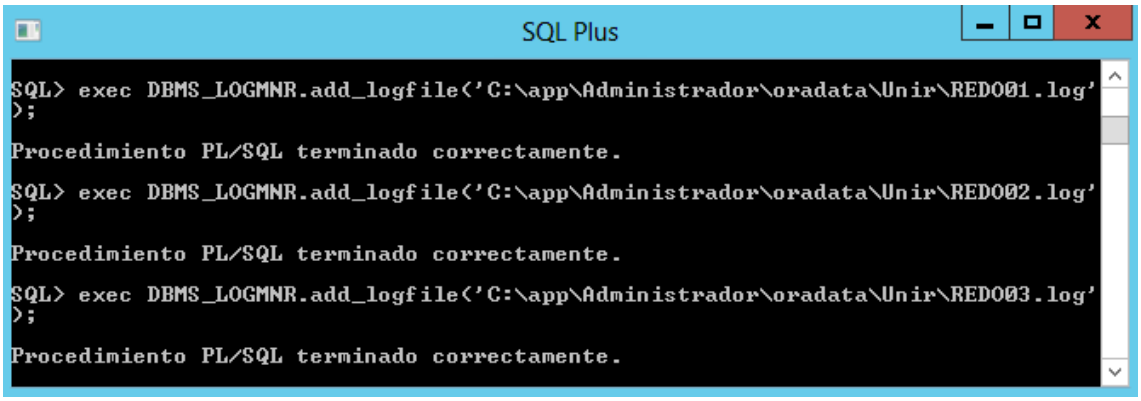


Figura 19. Carga Redo Logs en LogMiner Oracle

Y por ultimo cargamos el diccionario creado anteriormente

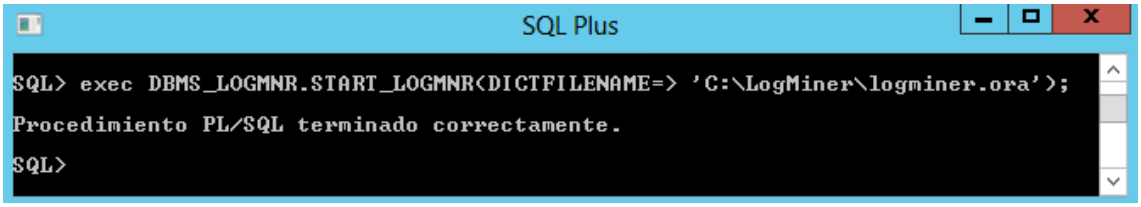


Figura 20. Carga diccionario LogMiner Oracle

3.4. Auditoría

3.4.1. MySQL. Auditoría

MySQL, a diferencia de otros gestores de bases de datos como Oracle o MS SQL server, no dispone de un servicio de auditoría como tal. Se pueden utilizar disparadores o triggers con ese fin, de tal manera que se guarden en una tabla auxiliar todas las modificaciones que se vayan realizando en todos los campos de una tabla de la base de datos.

A través de los trigger podremos saber que modificaciones se han realizado, con que usuario, en qué fecha y hora y que valores había antes y después de la modificación.

Para establecer la auditoría sobre la tabla “Productos”, crearemos una tabla auxiliar con los mismos campos para guardar el valor anterior (añadiendo el sufijo “_old”) y duplicaremos los campos para guardar el nuevo valor (añadiendo el sufijo “_new”)

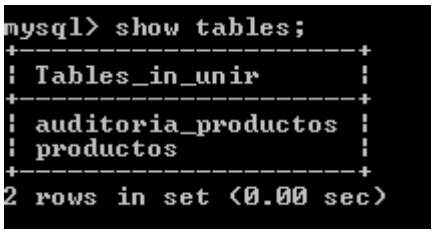


Figura 21. Tabla auditoría MySQL

Y a continuación creamos el trigger:

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER unir.auditoria_productos
    ->    AFTER UPDATE ON unir.productos
    ->    FOR EACH ROW
    -> BEGIN
    ->    INSERT INTO unir.auditoria_productos
    ->      (usuario, fecha, Producto_old,
    ->       Precio_old, Stock_old, Producto_new,
    ->       Precio_new, Stock_new)
    ->       VALUES (USER(), NOW(), OLD.Producto,
    ->       OLD.Precio, OLD.Stock, NEW.Producto,
    ->       NEW.Precio, NEW.Stock);
    -> END$$
```

Figura 22. Trigger MySQL

Con el comando AFTER_UPDATE indicamos que el trigger se ejecute cada vez que el usuario realice alguna modificación en la tabla “productos”

El trigger insertará un registro en la tabla auxiliar auditoría_productos con el valor anterior y el nuevo valor para cada campo, utilizando para ello las clausulas especiales OLD (el trigger

obtendrá el valor anterior al cambio del campo) y NEW (el trigger obtendrá el nuevo valor del campo).

La función USER() obtendrá y almacenara en la tabla el usuario actual de MySQL.

La función NOW() obtendrá la fecha y hora en que el usuario realiza el cambio en la tabla.

3.4.2. PostgreSQL. Auditoría

PostgreSQL, de la misma manera que MySQL y a diferencia de otros gestores de bases de datos como Oracle o MS SQL server, no dispone de un servicio de auditoría como tal. Se pueden utilizar disparadores o triggers con ese fin, de tal manera que se guarden en una tabla auxiliar todas las modificaciones que se vayan realizando en todos los campos de una tabla de la base de datos.

A través de los trigger podremos saber que modificaciones se han realizado, con que usuario, en qué fecha y hora y que valores había antes y después de la modificación.

Para establecer la auditoría sobre la tabla “Productos”, crearemos una tabla auxiliar en la que se registrarántodos los cambios realizados sobre la tabla

```
DROP TABLE auditoria_productos;
CREATE TABLE auditoria_productos (
  IdProducto serial NOT NULL,
  "NombreTabla" character(45) NOT NULL,
  "Operacion" char(10) NOT NULL,
  "ValorViejo" text,
  "ValorNuevo" text,
  "Fecha" timestamp without time zone NOT NULL,
  "Usuario" character(45) NOT NULL,
  "Host" character(45) NOT NULL,
  CONSTRAINT IdProducto PRIMARY KEY (IdProducto))
WITH (OIDS=FALSE);
ALTER TABLE auditoria_productos OWNER TO postgres;
```

Figura 23. Tabla auxiliar auditoría PostgreSQL

Creamos una función en PL/PgSQL que nos permite insertar los datos de aquellos registros que son afectados cada vez que se realiza una acción de tipo INSERT, UPDATE y DELETE en una tabla determinada, la cual es definida en la creación del Trigger

```
CREATE OR REPLACE FUNCTION fn_log_audit() RETURNS trigger AS
$$
BEGIN
  IF (TG_OP = 'DELETE') THEN
    INSERT INTO auditoria_productos ("NombreTabla", "Operacion", "ValorViejo", "ValorNuevo", "Fecha", "Usuario", "Host")
          VALUES (TG_TABLE_NAME, 'DELETE', OLD, NULL, now(), USER, inet_client_addr());
    RETURN OLD;
  ELSIF (TG_OP = 'UPDATE') THEN
    INSERT INTO auditoria_productos ("NombreTabla", "Operacion", "ValorViejo", "ValorNuevo", "Fecha", "Usuario", "Host")
          VALUES (TG_TABLE_NAME, 'UPDATE', OLD, NEW, now(), USER, inet_client_addr());
    RETURN NEW;
  ELSIF (TG_OP = 'INSERT') THEN
    INSERT INTO auditoria_productos ("NombreTabla", "Operacion", "ValorViejo", "ValorNuevo", "Fecha", "Usuario", "Host")
          VALUES (TG_TABLE_NAME, 'INSERT', NULL, NEW, now(), USER, inet_client_addr());
    RETURN NEW;
  END IF;
  RETURN NULL;
END;
$$
LANGUAGE 'plpgsql' VOLATILE COST 100;
ALTER FUNCTION fn_log_audit() OWNER TO postgres;
```

Figura 24. Función PostgreSQL

Y por ultimo creamos el Trigger en todas las tablas que queramos auditar, menos en la que se registrara la auditoría. Para este caso crearemos el trigger sobre la tabla productos, indicando que será ejecutado el trigger antes de la ejecución de una instrucción INSERT, UPDATE y DELETE para cada registro y le asignamos la función anterior.

```
CREATE TRIGGER tbl_atributos_tg_audit AFTER INSERT OR UPDATE OR DELETE
ON productos FOR EACH ROW EXECUTE PROCEDURE fn_log_audit();
```

Figura 25. Trigger PostgreSQL

Con el comando AFTER INSERT OR UPDATE OR DELETE indicamos que el trigger se ejecute cada vez que el usuario realice alguna modificación en la tabla “productos”

El trigger insertará un registro en la tabla auxiliar auditoria_productos con el valor anterior y el nuevo valor para cada campo, utilizando para ello las clausulas especiales OLD (el trigger obtendrá el valor anterior al cambio del campo) y NEW (el trigger obtendrá el nuevo valor del campo).

La variable USER el usuario actual de postgreSQL.

La función NOW() obtendrá la fecha y hora en que el usuario realiza el cambio en la tabla.

La función inet_client_addr() obtendrá y almacenara en la tabla el host desde el que se realiza la modificación.

3.4.3. MS SQL Server. Auditoría

La auditoría de una base de datos o de una instancia de SQL Server registra todos los eventos que se producen en el sistema. La auditoría se realiza a nivel de instancia y es posible tener varias auditorías por cada instancia.

Hay varios niveles de auditoría disponibles en SQL Server, dependiendo de los requisitos de cada instalación. Todas las ediciones de SQL Server admiten auditorías a nivel de servidor, las auditorías a nivel de base de datos se limitan a las ediciones Enterprise, Developer y Evaluation.

El registró y visualización de los cambios realizados en las distintas tablas de SQL Server se puede llevar a cabo a través de diferentes medios:

- Auditoría
- Triggers
- Change Data Capture (CDC)
- SQL Server Profiler

3.4.3.1. Mediante Auditoría

Creamos la auditoría para que se guarden los registros en disco

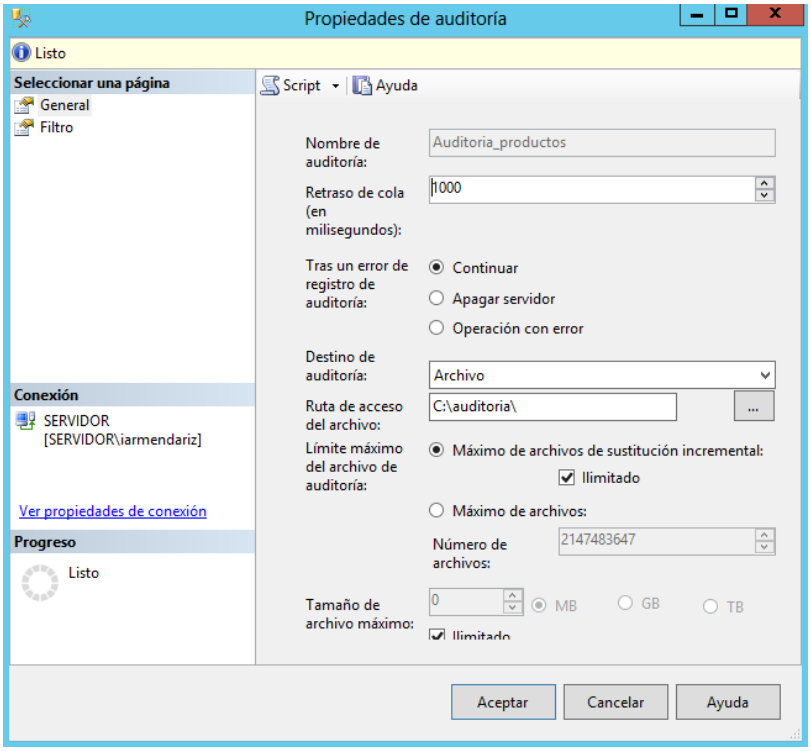


Figura 26. Propiedades auditoría MS SQL Server

Habilitamos la auditoría:

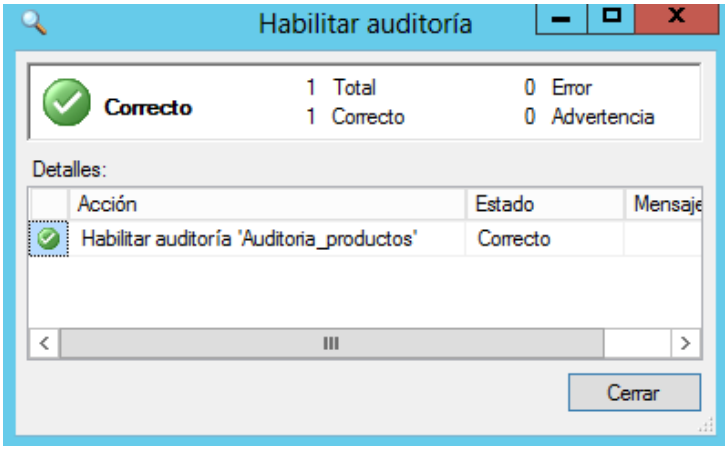


Figura 27. Habilitar auditoría MS SQL Server

Establecemos las operaciones que se auditarán:

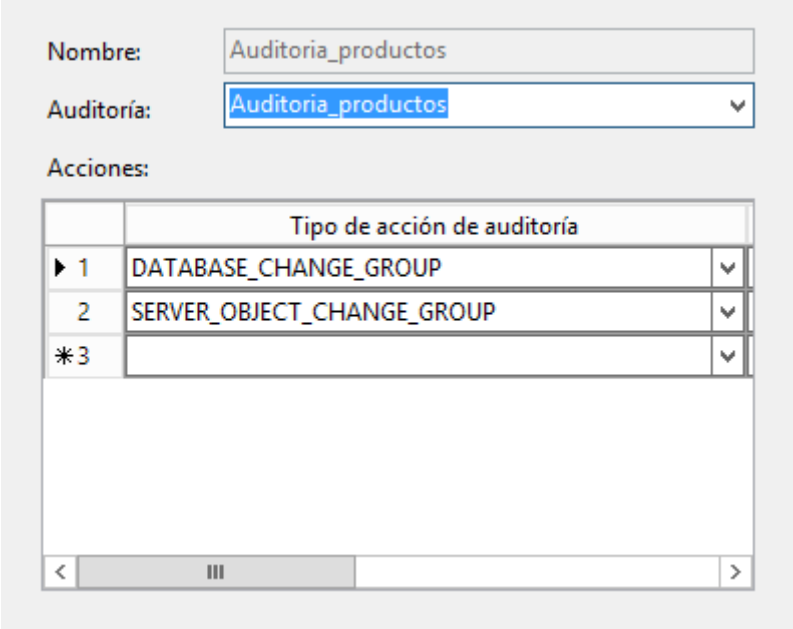


Figura 28. Operaciones auditoría MS SQL Server

3.4.3.2. Mediante Triggers

Lo primero que hay que hacer es crear la tabla de auditoría, sobre la cual el trigger escribirá el detalle de las transacciones realizadas:

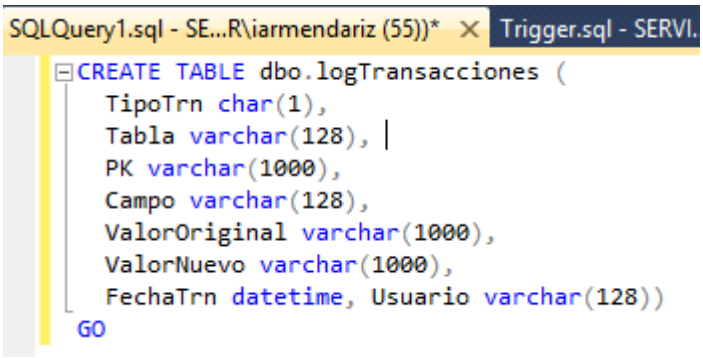


Figura 29. Tabla auxiliar MS SQL Server

En esta tabla se almacenara el tipo de transacción (I-Insert, U-Update, D-Delete), el nombre de la tabla, la clave del registro, los campos que han sufrido algún cambio, el valor original y el valor nuevo para las actualizaciones, la fecha en la que se hizo el cambio y el usuario que lo ejecutó.

A continuación creamos el desencadenador o trigger en la tabla productos.

```
CREATE TRIGGER dbo.triggerproductos ON dbo.productos FOR INSERT, UPDATE,
DELETE
AS

DECLARE @bit int ,
        @field int ,
        @maxfield int ,
        @char int ,
        @fieldname varchar(128) ,
        @TableName varchar(128) ,
        @PKCols varchar(1000) ,
        @sql varchar(2000),
        @UpdateDate varchar(21) ,
        @UserName varchar(128) ,
        @Type char(1) ,
        @PKSELECT varchar(1000)

SELECT @TableName = 'productos' --<-- cambiar el nombre de la tabla

-- Fecha y Usuario
SELECT @UserName = system_user ,
```

```
      @UpdateDate = convert(varchar(8), getdate(), 112) +
                  ' ' +
                  convert(varchar(12), getdate(), 114)

SET NoCount ON

-- Identificar que evento se está ejecutando (Insert, Update o Delete)
--en base a cursores especiales (inserted y deleted)
if exists (SELECT * FROM inserted)
  if exists (SELECT * FROM deleted) --Si es un update
    SELECT @Type = 'U'
  else                          --Si es un insert
    SELECT @Type = 'I'
else                            --si es un delete
    SELECT @Type = 'D'

-- Obtenemos la lista de columnas de los cursores
SELECT * INTO #ins FROM inserted
SELECT * INTO #del FROM deleted

-- Obtener las columnas de llave primaria
SELECT @PKCols = coalesce(@PKCols + ' and', ' on') +
      ' i.' +
      c.COLUMN_NAME + ' = d.' +
      c.COLUMN_NAME
 FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS pk
  JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE c
  ON c.TABLE_NAME = pk.TABLE_NAME
  AND c.CONSTRAINT_NAME = pk.CONSTRAINT_NAME
 WHERE pk.TABLE_NAME = @TableName AND
  pk.CONSTRAINT_TYPE = 'PRIMARY KEY'

-- Obtener la llave primaria y columnas para la inserción en la tabla de auditoría
SELECT
 @PKSELECT = coalesce(@PKSelect+'+','') +
 '''<' +
```

```
 COLUMN_NAME +
 '="+convert(varchar(100),coalesce(i.' +
 COLUMN_NAME +',d.' +
 COLUMN_NAME + '))+">'"
 FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS pk
 JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE c
  ON c.TABLE_NAME = pk.TABLE_NAME
  AND c.CONSTRAINT_NAME = pk.CONSTRAINT_NAME
 WHERE pk.TABLE_NAME = @TableName
  AND CONSTRAINT_TYPE = 'PRIMARY KEY'

if @PKCols is null --<-- Este trigger solo funciona si la tabla tiene llave primaria
 BEGIN
  RAISERROR('no PK on table %s', 16, -1, @TableName)
  RETURN
 END

--Loop para armar el query de inserción en la tabla de log.
--Un registro por cada campo afectado.
SELECT
 @field = 0,
 @maxfield = max(ORDINAL_POSITION)
 FROM INFORMATION_SCHEMA.COLUMNS
 WHERE TABLE_NAME = @TableName

while @field < @maxfield
 BEGIN
  SELECT @field = min(ORDINAL_POSITION)
   FROM INFORMATION_SCHEMA.COLUMNS
   WHERE TABLE_NAME = @TableName and ORDINAL_POSITION > @field
  SELECT @bit = (@field - 1 )% 8 + 1
  SELECT @bit = power(2,@bit - 1)
  SELECT @char = ((@field - 1) / 8) + 1
  if substring(COLUMNS_UPDATED(),@char, 1) & @bit > 0 or @Type in ('I','D')
   BEGIN
    SELECT @fieldname = COLUMN_NAME
```

```
    FROM INFORMATION_SCHEMA.COLUMNS
        WHERE TABLE_NAME = @TableName and ORDINAL_POSITION = @field
   SELECT @sql = 'insert LogTransacciones (TipoTrn, Tabla, PK, Campo, ValorOriginal, ValorNuevo, FechaTrn, Usuario)'
   SELECT @sql = @sql +  ' SELECT ''' + @Type + ''''
   SELECT @sql = @sql +  ',''' + @TableName + ''''
   SELECT @sql = @sql +  ',' + @PKSelect
   SELECT @sql = @sql +  ',''' + @fieldname + ''''
   SELECT @sql = @sql +  ',convert(varchar(1000),d.' + @fieldname + ')'
   SELECT @sql = @sql +  ',convert(varchar(1000),i.' + @fieldname + ')'
   SELECT @sql = @sql +  ',''' + @UpdateDate + ''''
   SELECT @sql = @sql +  ',''' + @UserName + ''''
   SELECT @sql = @sql +  ' from #ins i full outer join #del d'
   SELECT @sql = @sql +  @PKCols
   SELECT @sql = @sql +  ' where i.' + @fieldname + ' <> d.' + @fieldname
   SELECT @sql = @sql +  ' or (i.' + @fieldname + ' is null and  d.' + @fieldname + ' is not null)'
   SELECT @sql = @sql +  ' or (i.' + @fieldname + ' is not null and  d.' + @fieldname + ' is null)'
    exec (@sql)
  END
 END

SET NoCount OFF
GO
```

Figura 30. Trigger MS SQL Server

3.4.3.3. Mediante SQL Server Profiler

SQL Server Profiler sirve para monitorear los eventos seleccionados, bien en tiempo real o bien sobre archivos grabados.

Rastreara y registrara todos los eventos que le indiquemos. En este caso únicamente vamos a rastrear transacciones realizadas sobre las tablas.

Creamos un nuevo seguimiento con la plantilla en blanco

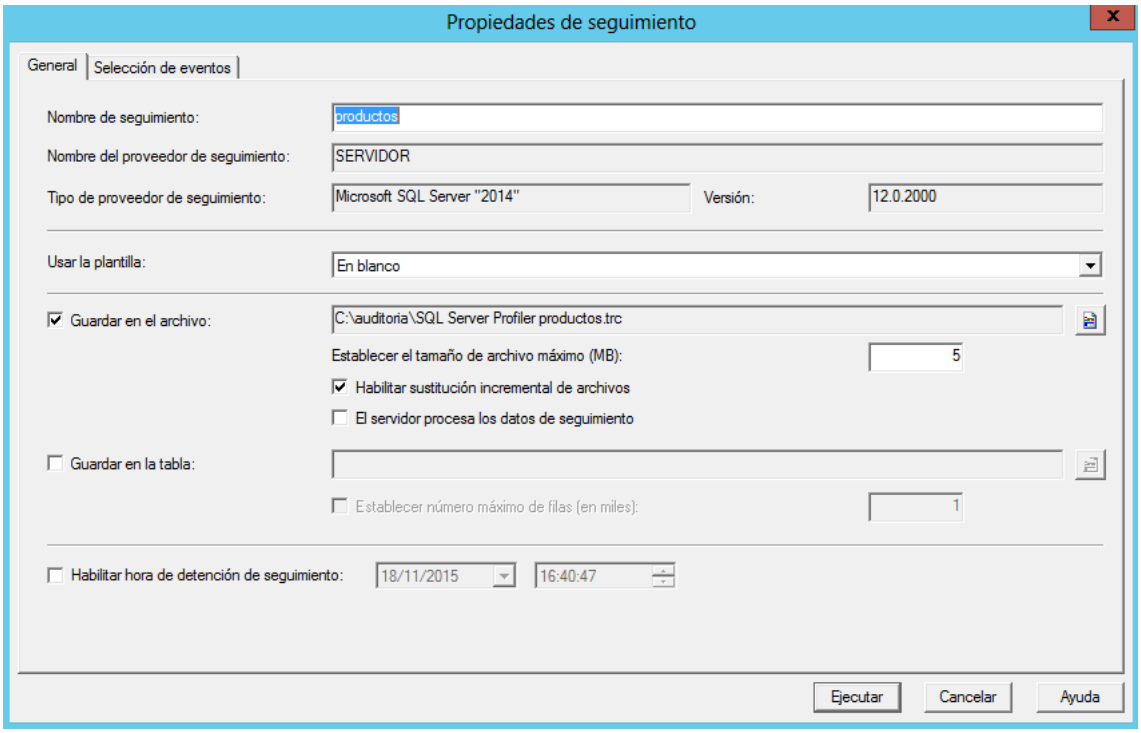


Figura 31. Propiedades seguimiento MS SQL Server

Seleccionamos los eventos a rastrear:

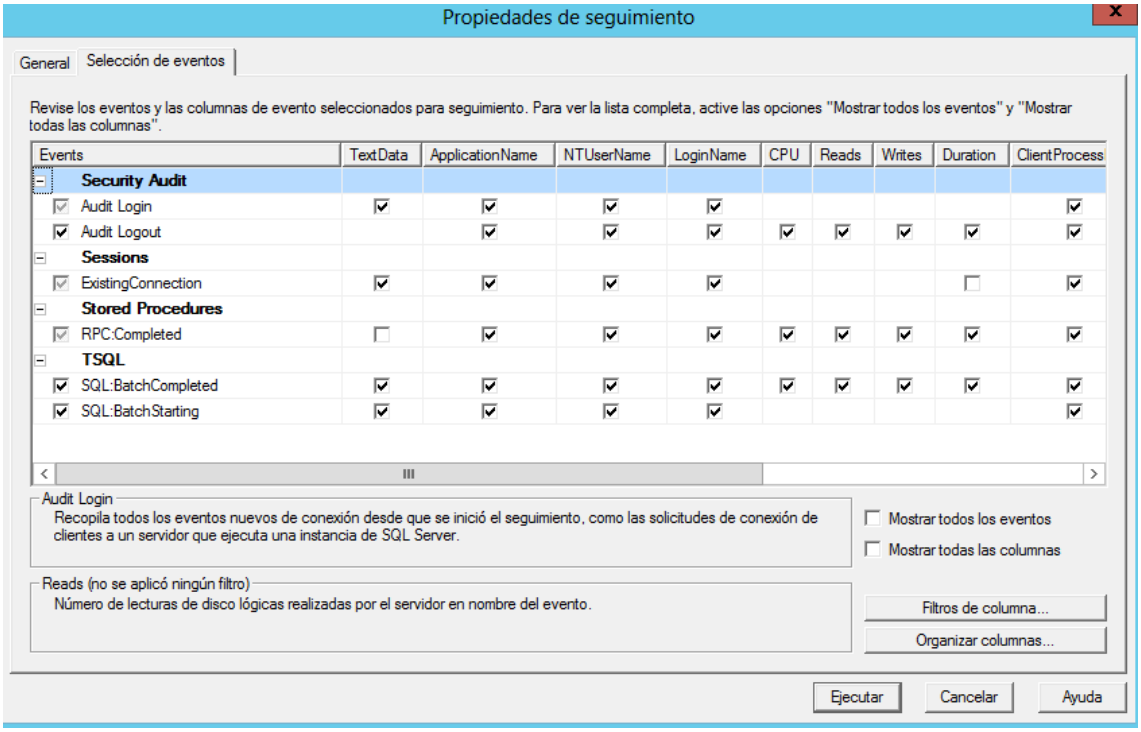


Figura 32. Propiedades seguimiento MS SQL Server

3.4.4. Oracle. Auditoría

En el caso de Oracle, la auditoría permite al administrador hacer un seguimiento del uso de la base de datos.

Tenemos dos tipos de auditoría principal en Oracle, la auditoría estándar (Standard Auditing) y la auditoría de grano fino FGA (Fine Grained Auditing), que extiende la auditoría estándar y, además, captura la sentencia SQL que ha sido ejecutada.

Standard auditing nos permite tener un registro de información acerca de una operación hecha en base de datos (información como usuario de ejecución, fecha, terminal). Esta opción de auditoría está incluida dentro de la licencia Standard Edition y Enterprise Edition.

FGA nos permite registrar operaciones en base a políticas de auditoría, como acciones sobre campos específicos, horarios de ejecución, y condiciones de una ejecución. Esta opción de auditoría está incluida dentro de la licencia Enterprise Edition

Además de las auditorías citadas, se pueden crear triggers sobre las tablas para llevar el registro de los cambios realizados sobre ellas.

3.4.4.1. Auditoría Estándar

La información de las auditorías se almacena en el diccionario de datos, en la tabla SYS.AUD$.

Se pueden definir tres tipos de acciones:

- Intentos de inicio de sesión
- Accesos a objetos
- Acciones de la base de datos

Cuando se realizan auditorías, la funcionalidad de la base de datos es dejar constancia de los comandos correctos e incorrectos. Esto puede modificarse cuando se configura cada tipo de auditoría.

En Oracle 12, la auditoría viene activada por defecto

Para auditar las sesiones:

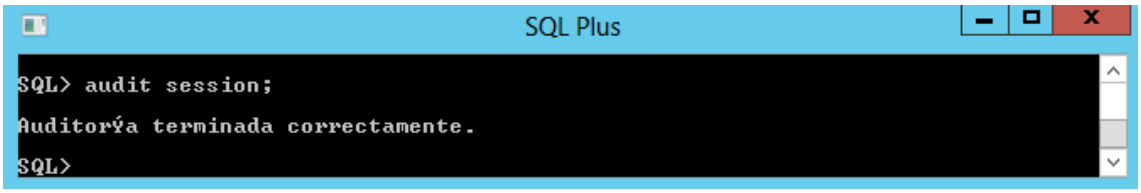


Figura 33. Auditar sesiones Oracle

Para auditar la tabla:

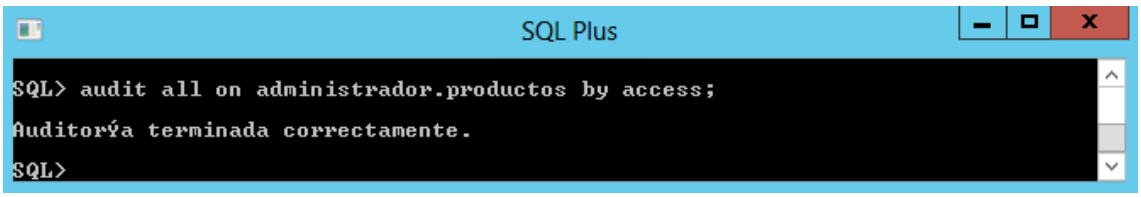


Figura 34. Auditar tabla Oracle

3.4.4.2. Fine Grained Auditing (FGA)

Este tipo de auditoría nos permite registrar operaciones en base a políticas. Lo primero será establecer las políticas sobre las tablas que queramos auditar

Para activar una política tendremos que indicar la tabla y las acciones que queramos auditar

EXEC DBMS_FGA.ADD_POLICY (object_name=>'productos', policy_name=>'AuditoriaPrecio', audit_column=>'Precio', statement_types=>'select, update, insert');

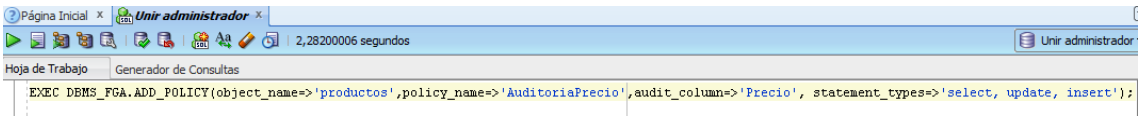


Figura 35. Política FGA Oracle

Se pueden visualizar todas las políticas activas con el siguiente comando:

SELECT object_schema, object_name,policy_name, policy_column, enabled, sel, ins, upd, del FROM dba_audit_policies;

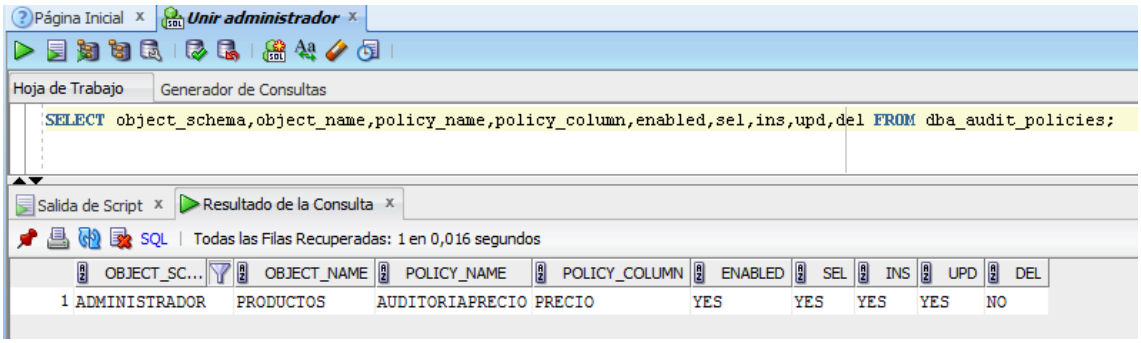


Figura 36. Visualizar políticas FGA Oracle

Y por último, para visualizar todos los registros auditados con la política usaremos:

SELECT TIMESTAMP, db_user, object_schema, object_name, policy_name, sql_text FROM DBA_FGA_AUDIT_TRAIL ORDER BY TIMESTAMP DESC;

3.5. Ataques

Se realizaráncuatro tipos de ataque distintos sobre los cuatro gestores de base de datos. Los ataques a realizar son:

- Ataque interno: Usuario administrador que modifica datos.
- Ataque interno: Usuario de la empresa que modifica datos.
- Ataque externo: SQL Injection.
- Ataque externo: Fuerza bruta.

3.5.1. Ataque 1: Usuario administrador que modifica datos

En este primer ataque, un usuario con privilegios de administrador modificara datos de la tabla productos. El usuario con privilegios de administrador modifica el precio del producto Ladrillo de 13€ a 17€:

- **MySQL.** El usuario root modifica el precio del producto Ladrillo mediante la consola de MySQL.

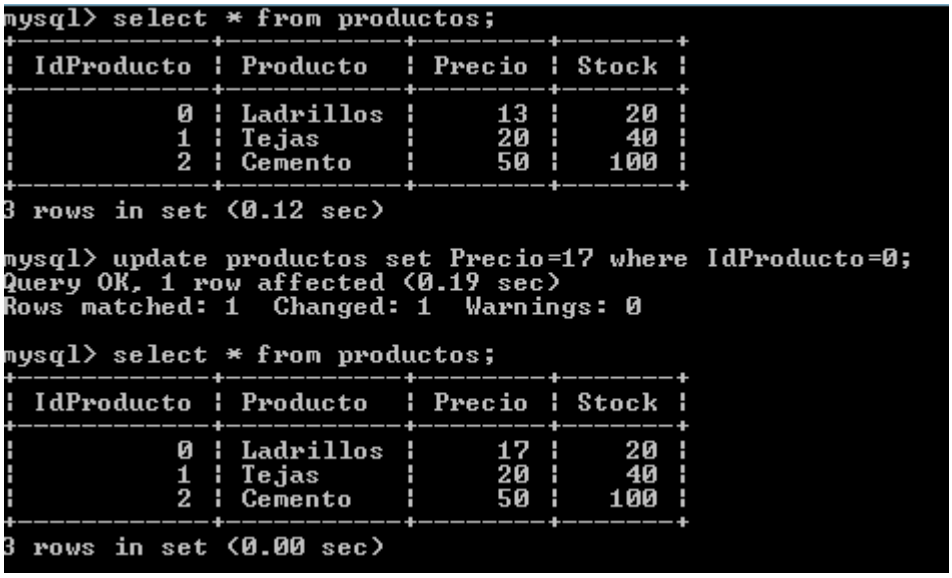
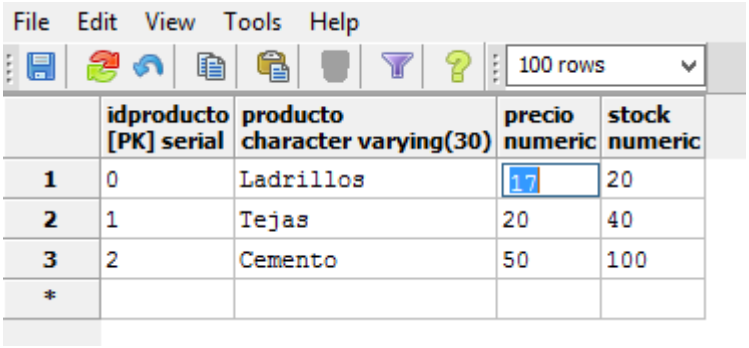


Figura 37. Ataque 1 MySQL

- **PostgreSQL.** El usuario root modifica el precio del producto Ladrillo mediante la herramienta pgAdmin III de PostgreSQL.



	idproducto [PK] serial	producto character varying(30)	precio numeric	stock numeric
1	0	Ladrillos	17	20
2	1	Tejas	20	40
3	2	Cemento	50	100
*				

Figura 38. Ataque 1 PostgreSQL

- **MS SQL Server.** El usuario administrador modifica el precio del producto Ladrillo mediante Microsoft SQL Server Management Studio.

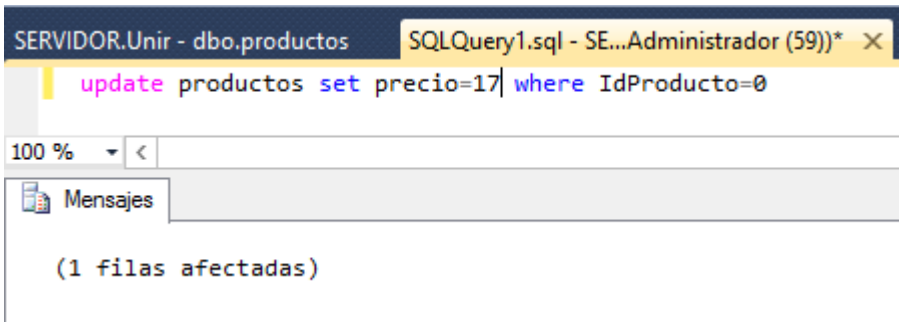


Figura 39. Ataque 1 MS SQL Server

- **Oracle.** El usuario administrador modifica el precio del producto Ladrillo mediante la herramienta SQL Plus

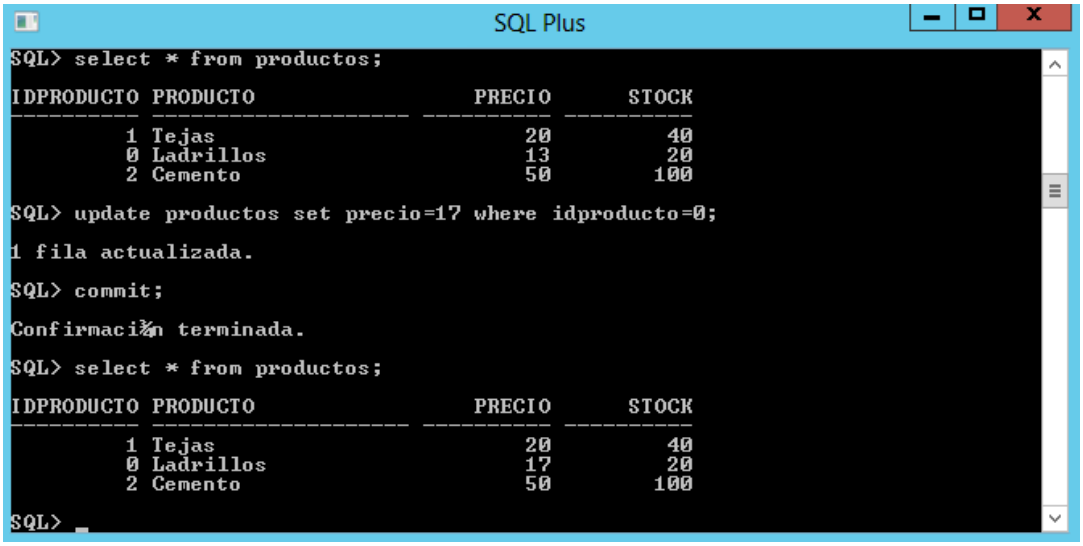


Figura 40. Ataque 1 Oracle

La dirección de la empresa detectará que el precio con el que se está vendiendo un producto no es el correcto y querrá averiguar quién y cuándo lo ha modificado para tomar las acciones oportunas.

3.5.2. Ataque 2: Usuario de la empresa que modifica datos

En este segundo ataque, un usuario normal de la empresa modificara datos de la tabla productos mediante una aplicación de escritorio.

Para emular una aplicación de escritorio he vinculado las tablas de los cuatro SGBD (MySQL, PostgreSQL, MS SQL Server y Oracle) sobre una aplicación en Access 2007. Mediante un formulario de Access y desde una maquina de la red local, el usuario "iarmendariz" modifica el precio del producto Tejas de 20€ a 30€:

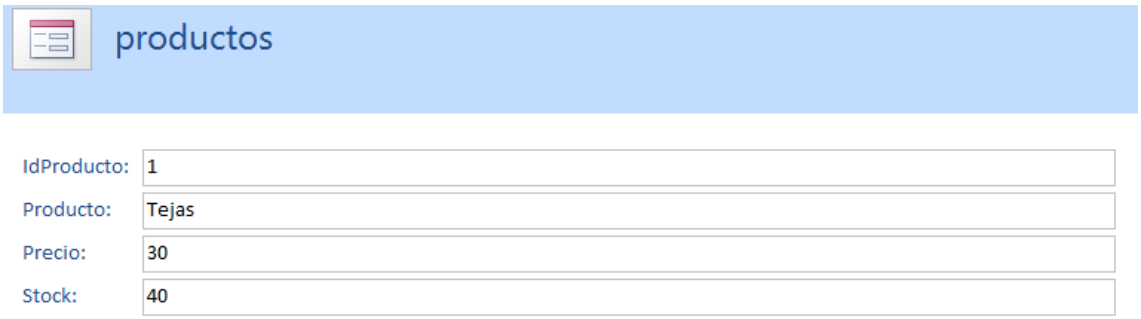


Figura 41. Ataque 2

La dirección de la empresa detectará que el precio con el que se está vendiendo un producto no es el correcto y querrá averiguar quién y cuándo lo ha modificado para tomar las acciones oportunas.

3.5.3. Ataque 3: SQL Injection

En este tercer ataque se realizara un ataque de SQL Injection. Este tipo de ataque es una técnica donde un atacante crea o altera comandos SQL existentes para exponer datos ocultos, sobrescribir datos, o ejecutar comandos peligrosos a nivel de sistema en el equipo que hospeda la base de datos.

Para emular una aplicación sobre la que realizar un ataque de SQL Injection, he vinculado las tablas de los cuatro SGBD sobre una aplicación en Access 2007. He creado un formulario en el que el usuario puede introducir el código del producto y la aplicación devuelve el nombre, precio y stock del mismo realizando una select de la tabla para ello.

La sentencia SQL existente en la base de datos es:

- **MySQL**

```
strSQL1 = "Select * from productos where idproducto= " & Me.Texto
Set rst1 = dbs1.OpenRecordset(strSQL1)
EtiProducto.Caption = rst1!Producto
EtiPrecio.Caption = rst1!precio
EtiStock.Caption = rst1!Stock
```

Figura 42. Ataque 3 MySQL

- **PostgreSQL**

```
strSQL1 = "Select * from public_productos where idproducto= " & Me.Texto
Set rst1 = dbs1.OpenRecordset(strSQL1)
EtiProducto.Caption = rst1!Producto
EtiPrecio.Caption = rst1!precio
EtiStock.Caption = rst1!Stock
```

Figura 43. Ataque 3 PostgreSQL

- **MS SQL Server**

```
strSQL1 = "Select * from dbo_productos where idproducto= " & Me.Texto
Set rst1 = dbs1.OpenRecordset(strSQL1)
EtiProducto.Caption = rst1!Producto
EtiPrecio.Caption = rst1!Precio
EtiStock.Caption = rst1!Stock
```

Figura 44. Ataque 3 MS SQL Server

- **Oracle**

```
Dim dbs1 As DAO.Database
Dim rst1 As DAO.Recordset
Dim strSQL1 As String
Set dbs1 = CurrentDb
strSQL1 = "Select * from administrador_productos where idproducto= " & Me.Texto
Set rst1 = dbs1.OpenRecordset(strSQL1)
EtiProducto.Caption = rst1!Producto
EtiPrecio.Caption = rst1!Precio
EtiStock.Caption = rst1!Stock
```

Figura 45. Ataque 3 Oracle

Figura 46. Ataque 3

Para realizar el ataque de SQL Injection voy a aprovechar la vulnerabilidad existente en el código de la aplicación en la que no se valida la entrada. El atacante introduciría un código de producto inexistente por no conocer los productos de la tabla, pero alterando la sentencia SQL.

Como se puede apreciar en la imagen siguiente, alterando la sentencia incluyendo 1=1 la aplicación muestra los datos del primer producto de la tabla

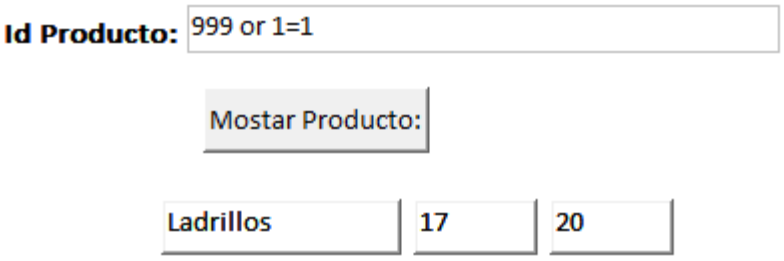


Figura 47. Ataque 3 resultado

La dirección de la empresa detectará que desde fuera de la organización se está accediendo a la base de datos y querrá averiguar quién y cuándo ha accedido para tomar las acciones oportunas.

3.5.4. Ataque externo: Fuerza bruta

En este cuarto ataque se realizara un ataque de fuerza bruta. Se denomina ataque de fuerza bruta a la forma de recuperar una clave probando todas las combinaciones posibles hasta encontrar aquella que permite el acceso.

Para realizar este tipo de ataque voy a utilizar un equipo con Linux. Mediante la herramienta nmap escanearemos los puertos abiertos de los servidores donde residen las bases de datos y una vez localizado el puerto del SGBD utilizare la herramienta Hydra para realizar el ataque de fuerza bruta.

- **MySQL**

Utilizando nmap vemos que el puerto TCP 3306 de MySQL está abierto

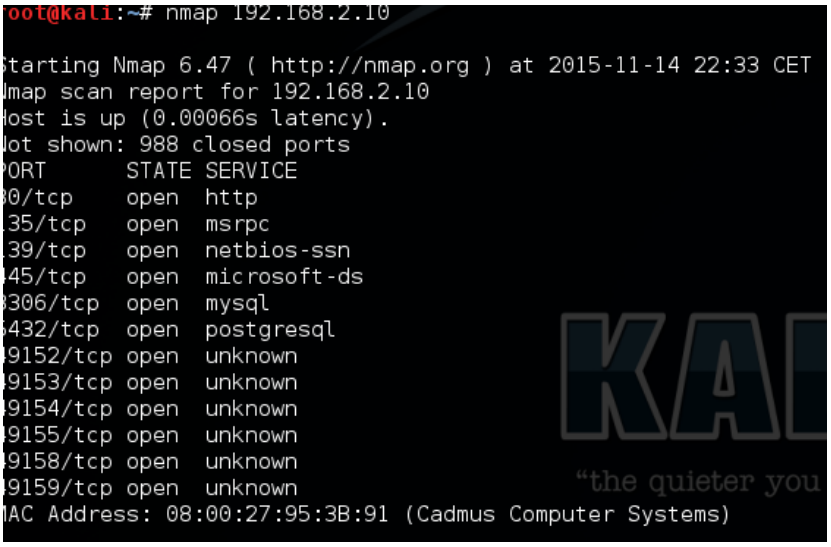


Figura 48. Nmap MySQL

Ahora, mediante la herramienta Hydra y utilizando un diccionario de contraseñas para el usuario root realizamos el ataque de fuerza bruta sobre el servidor MySQL

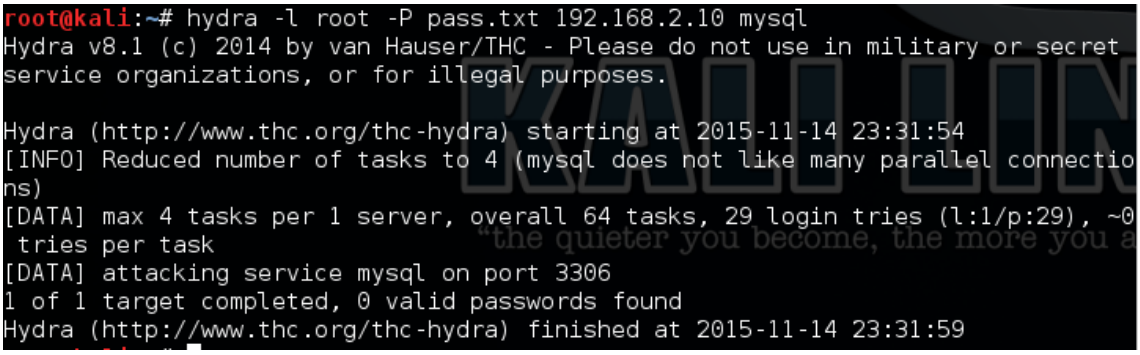


Figura 49. Ataque 4 Hydra MySQL

- **PostgreSQL**

Utilizando nmap vemos que el puerto TCP 5432 de PostgreSQL está abierto

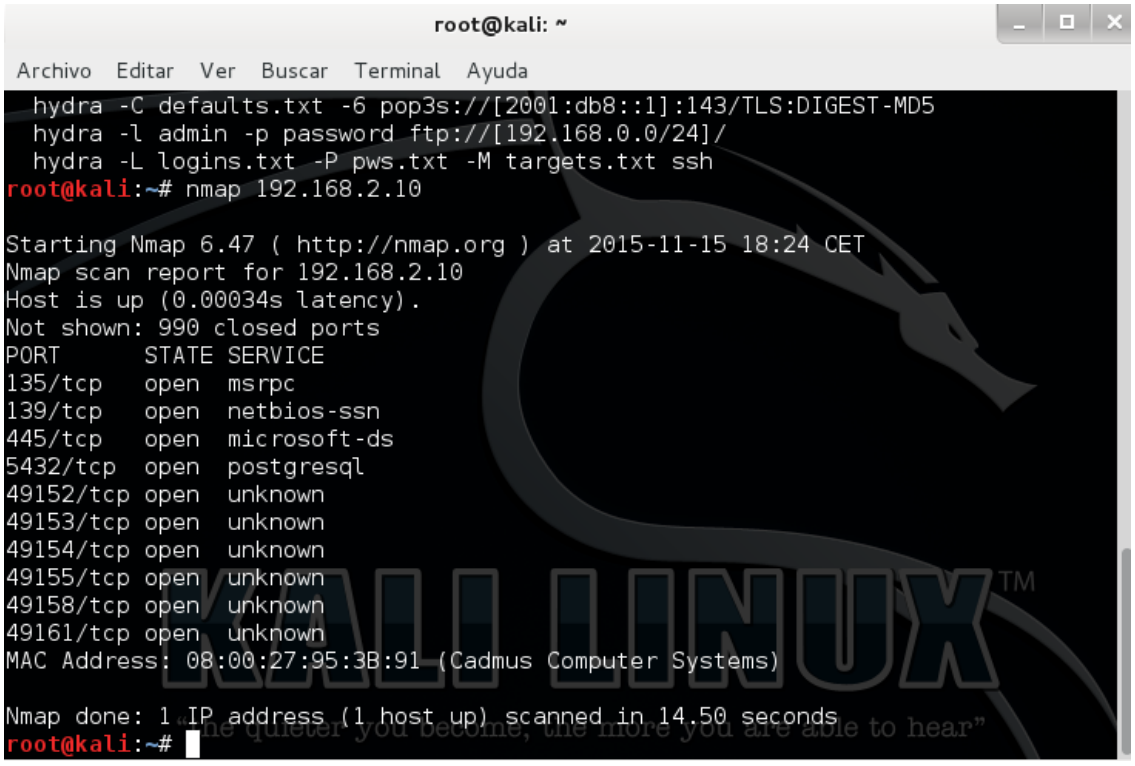


Figura 50. Nmap PostgreSQL

Ahora, mediante la herramienta Hydra y utilizando un diccionario de contraseñas para el usuario postgres realizamos el ataque de fuerza bruta sobre el servidor PostgreSQL

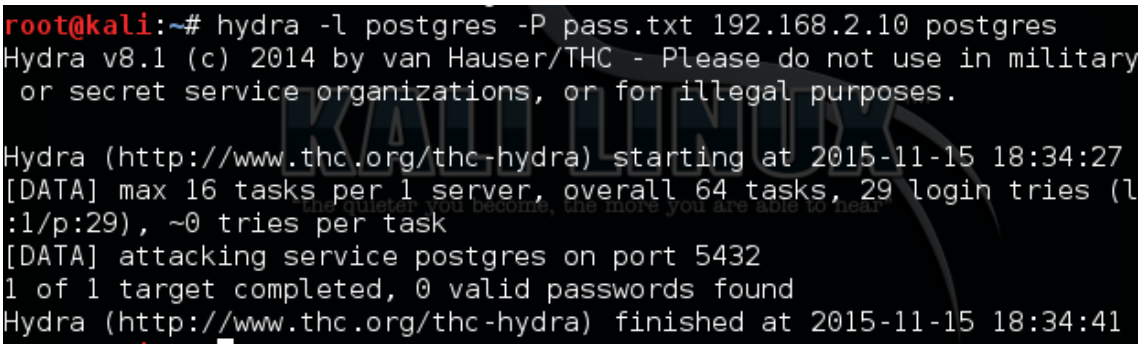


Figura 51. Ataque 4 Hydra PostgreSQL

- **MS SQL Server**

Utilizando nmap vemos que el puerto TCP 1433 de SQL Server está abierto

Figura 52. Nmap MS SQL Server

Ahora, mediante la herramienta Hydra y utilizando un diccionario de contraseñas para el usuario “sa” realizamos el ataque de fuerza bruta sobre el servidor SQL Server.

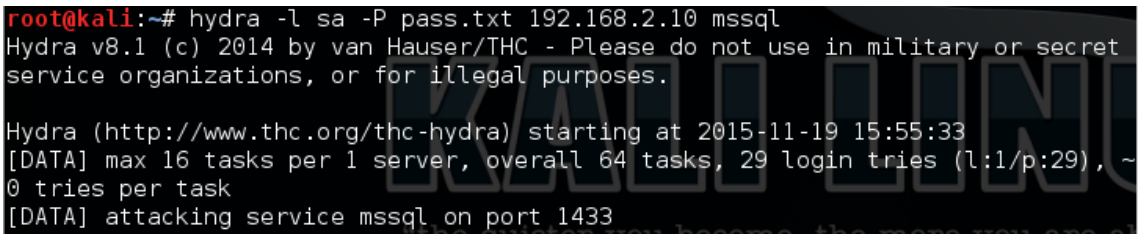


Figura 53. Ataque 4 Hydra MS SQL Server

- **Oracle**

Utilizando nmap vemos que el puerto TCP 1521 de Oracle está abierto

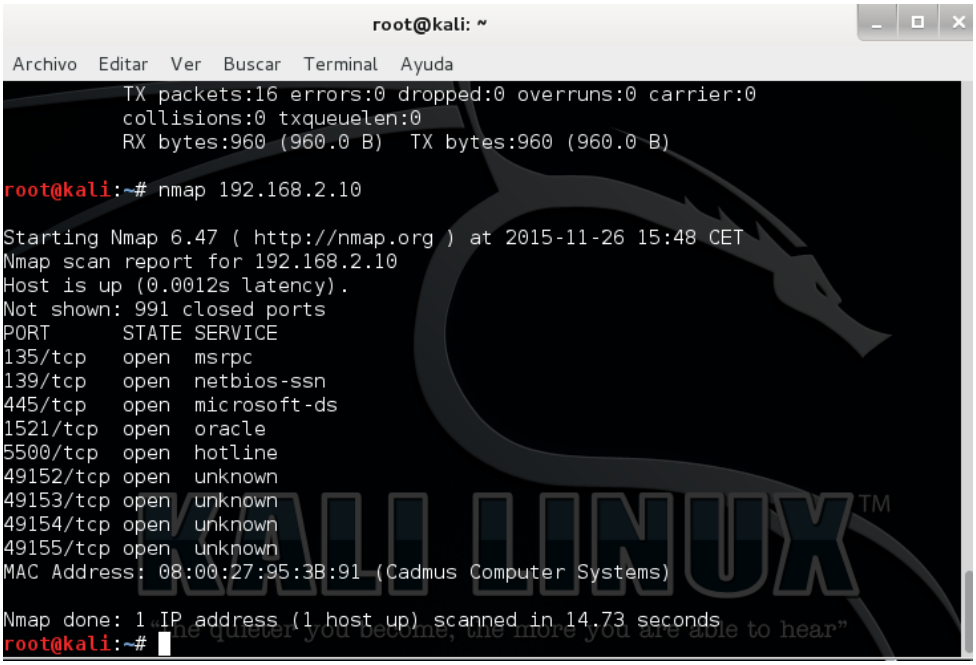


Figura 54. Nmap Oracle

Ahora, mediante la herramienta Oracle Brute Force y utilizando un diccionario de contraseñas para el usuario system realizamos el ataque de fuerza bruta sobre el servidor Oracle. En este caso utilizamos una herramienta distinta a Hydra porque esta última nos estaba dando problemas.

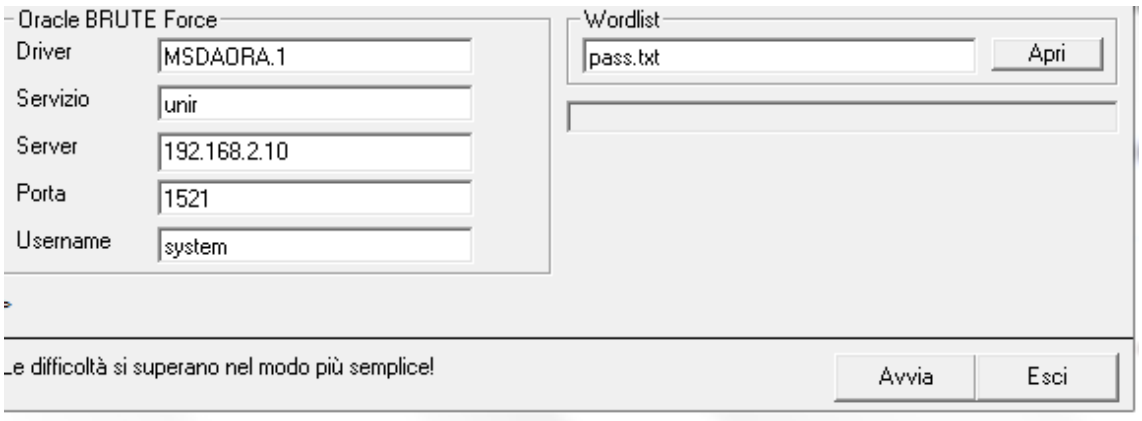


Figura 55. Ataque 4 Brute force Oracle

3.6. Adquisición de evidencias

Partimos de un escenario en el que tenemos se van a realizar cuatro ataques sobre la base de datos y en el que bastante claro que es lo que tenemos que buscar y donde. Un escenario en el que no es necesario adquirir las evidencias volátiles (memoria RAM, usuarios activos, cache, etc.) puesto que desde que ocurrieron los ataques el servidor se ha reiniciado más de una vez.

Adquirir una evidencia es crear un archivo que contenga toda la información contenida en la evidencia digital. Incluidos los espacios marcados como vacios.

En el caso de este escenario vamos a realizar una adquisición en caliente, con el servidor encendido, puesto que el mismo sigue dando servicio a los usuarios y no se puede parar. Utilizaremos para ello la herramienta AccessData FTK Imager, que permite la adquisición física de los dispositivos conectados al equipo.

FTK Imager de AccessData, es una herramienta para realizar réplicas y visualización previa de datos con licencia gratuita. Hemos seleccionado esta herramienta por ser de libre uso y sencillo su manejo. Mediante esta herramienta realizaremos una imagen del disco del servidor en formato RAW.

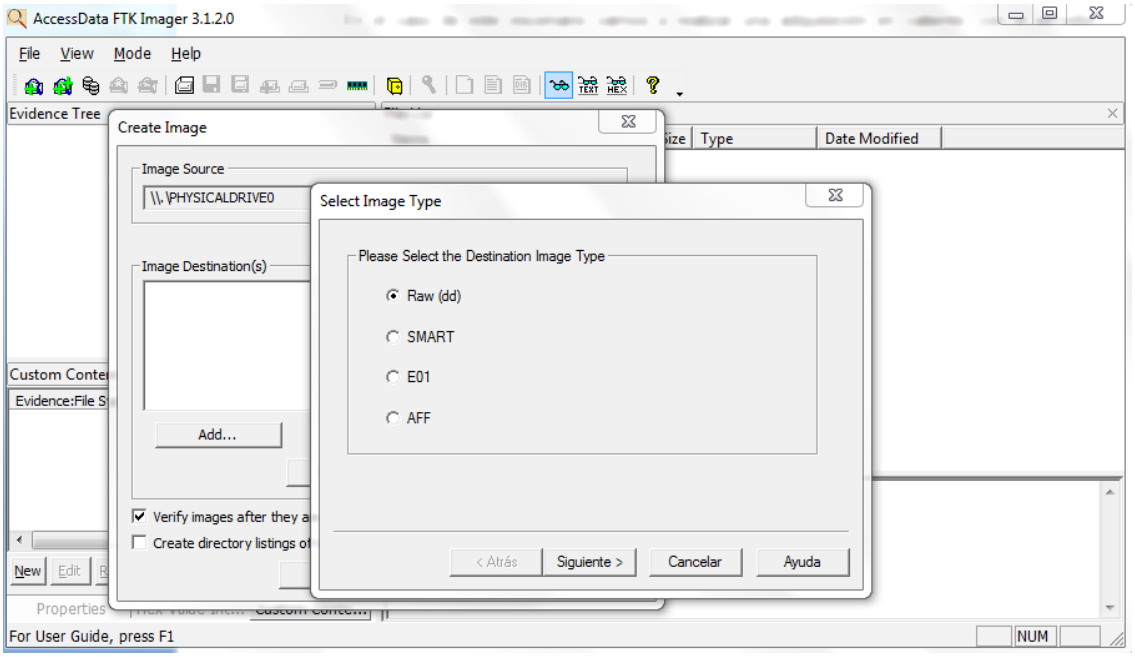


Figura 56. FTK Imager

El formato RAW almacena la información tal cual está en la propia evidencia original, sin compresión ni metadatos. Una vez adquirida la evidencia, podremos montarla para analizarla e incluso arrancarla mediante herramientas de virtualización como VirtualBox.

3.7. Análisis MySQL

Como se ha comentado anteriormente, el análisis se realizara sobre dos modos diferentes del gestor de base de datos, por un lado cuando únicamente se registran logs y por otro cuando además existe establecida una auditoría sobre la base de datos.

3.7.1. Análisis ataque 1

3.7.1.1. Análisis sobre SGBD sin auditoría

En este caso el análisis se tiene que realizar sobre los logs generados por el propio gestor. Dependiendo de la instalación, estos logs podrán situarse en diferentes lugares del disco.

Como ya se ha comentado anteriormente, MySQL genera varios logs en los que se registra su actividad.

En el escenario que nos ocupa, necesitamos saber quién y cuándo modifico el precio de un producto, para lo cual buscaremos y analizaremos el registro “General query log”. En este log se registran las conexiones establecidas con el servidor y las declaraciones o sentencias realizadas sobre la base de datos.

Una vez localizado el log en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se modifica el precio al producto.

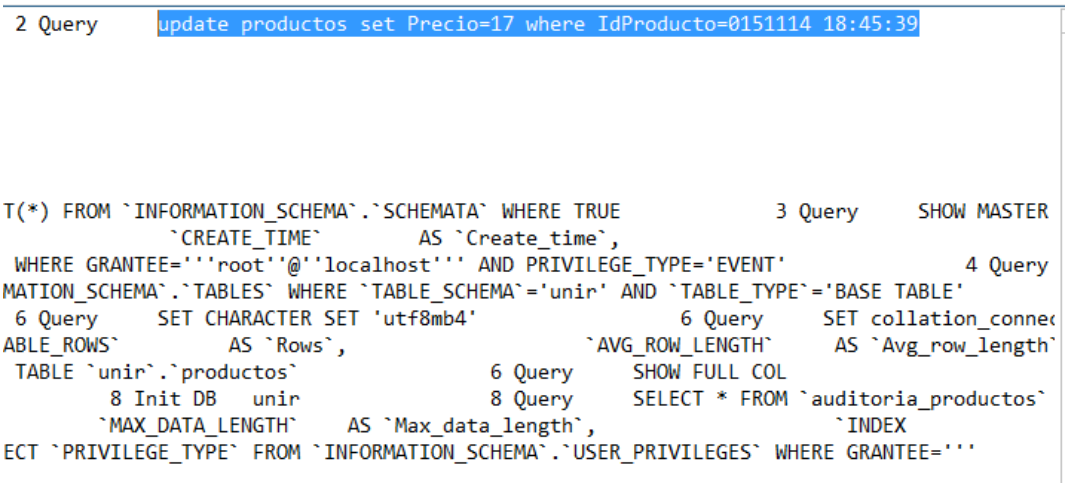


Figura 57. Log ataque 1 MySQL

En la imagen anterior se puede observar que la modificación del precio del producto con IdProducto=0, que corresponde al producto “Ladrillo”, se modifico a 17€ el día 14/11/2015 a las 18:45:39.

Lo siguiente será averiguar quién realizo esa modificación, y analizando ese mismo log podremos ver que lo realizo el usuario root desde la misma máquina que contiene la base de datos, a la cual se conecto el día 14/11/2015 a las 18:44:34.

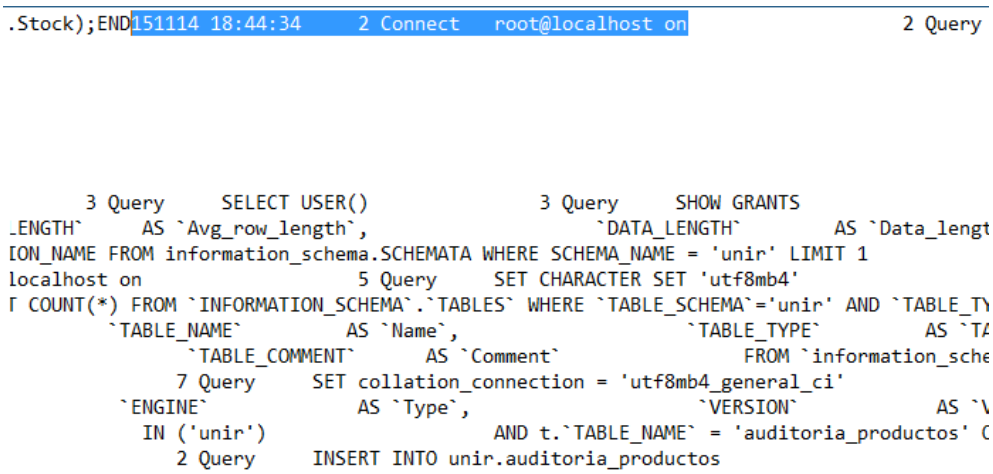


Figura 58. Log ataque 1 MySQL

3.6.1.2. Análisis sobre SGBD con auditoría

Para este segundo modo en el que existe una auditoría de la base de datos, y una vez que hemos recopilado toda la información sobre cómo está establecida esa auditoría por parte de los administradores de sistemas de la empresa, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

En este caso, todas las transacciones realizadas sobre la tabla productos se auditan sobre la tabla auditoria_productos.

Analizando esta tabla podremos comprobar que el usuario root, desde la misma máquina de la base de datos (localhost), modifico el precio del producto Ladrillo de 13€ a 17€ el día 14/11/2015 a las 18:45:38.

IdProducto	usuario	fecha	Producto_old	Precio_old	Stock_old	Producto_new	Precio_new	Stock_new
1	root@localhost	2015-11-14 17:20:54	Cemento	50	100	Cemento	50	101
2	root@localhost	2015-11-14 17:33:39	Cemento	50	101	Cemento	50	100
3	root@localhost	2015-11-14 17:35:36	Cemento	50	100	Cemento	50	1001
4	root@localhost	2015-11-14 17:36:52	Cemento	50	1001	Cemento	50	102
5	root@localhost	2015-11-14 17:37:31	Cemento	50	102	Cemento	50	1022
6	root@localhost	2015-11-14 17:38:07	Cemento	50	1022	Cemento	50	10223
7	root@localhost	2015-11-14 17:42:48	Cemento	50	10223	Cemento	50	1022
8	root@localhost	2015-11-14 17:47:02	Cemento	50	1022	Cemento	50	102233
9	root@localhost	2015-11-14 17:47:48	Cemento	50	102233	Cemento	50	1022340
10	root@localhost	2015-11-14 17:58:47	Tejas	20	40	Tejas	20	99940
11	root@localhost	2015-11-14 18:11:23	Ladrillos	13	20	Ladrillos	13	201
12	iarmendariz@P-ARMENDARIZ	2015-11-14 18:24:17	Ladrillos	13	201	Ladrillos	13	2018
13	root@localhost	2015-11-14 18:39:52	Ladrillos	13	2018	Ladrillos	13	20
14	root@localhost	2015-11-14 18:40:00	Tejas	20	99940	Tejas	20	40
15	root@localhost	2015-11-14 18:40:06	Cemento	50	1022340	Cemento	50	100
16	root@localhost	2015-11-14 18:45:38	Ladrillos	13	20	Ladrillos	17	20

Figura 59. Auditoría ataque 1 MySQL

3.7.2. Análisis ataque 2

3.7.2.1. Análisis sobre SGBD sin auditoría

En este caso el análisis se tiene que realizar sobre los logs generados por el propio gestor.

MySQL genera varios logs en los que se registra su actividad. En el escenario que nos ocupa, necesitamos saber quién y cuándo modifico el precio de un producto, para lo cual buscaremos y analizaremos el registro “General query log”. En este log se registran las conexiones establecidas con el servidor y las declaraciones o sentencias realizadas sobre la base de datos.

Una vez localizado el log en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se modifica el precio al producto.

```
        12 Connect   iarmendariz@P-ARMENDARIZ on unir                12 Query     SET NAMES utf8
FROM `productos`  WHERE `IdProducto` = '1' OR `IdProducto` = '2' OR `IdProducto` = '2' OR `IdProducto` = '2'


UPDATE `productos` SET `Precio`='3.0000000000000000e+001'  WHERE `IdProducto` = '1' AND `Producto` = 'Tejas'
```

Figura 60. Log ataque 2 MySQL

En la imagen anterior se puede observar que la modificación del precio del producto con IdProducto=1, que corresponde al producto “Tejas”, se modifico a 30€.

En la siguiente imagen se observa que la modificación se realizo el día 14/11/2015 a las 20:07:07.

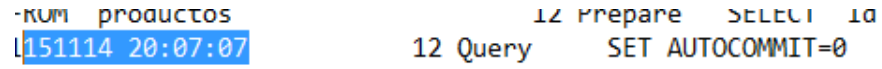


Figura 61. Log ataque 2 MySQL

Lo siguiente será averiguar quién realizo esa modificación, y analizando el log podemos ver que lo realizo el usuario iarmendariz desde la maquina P-ARMENDARIZ, a la cual se conecto el día 14/11/2015 a las 20:05:43.

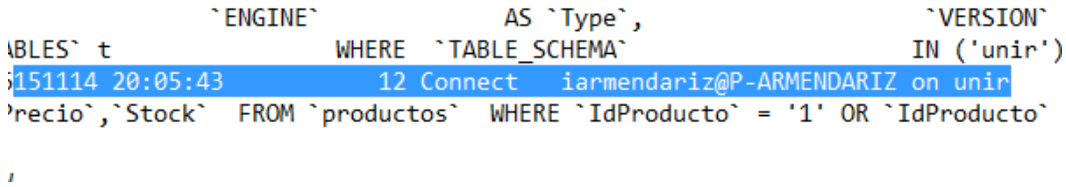


Figura 62. Log ataque 2 MySQL

3.6.2.2. Análisis sobre SGBD con auditoría

Para este segundo modo en el que existe una auditoría de la base de datos, y una vez que hemos recopilado toda la información sobre cómo está establecida esa auditoría por parte de los administradores de sistemas de la empresa, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

En este caso, todas las transacciones realizadas sobre la tabla productos se auditan sobre la tabla auditoria_productos.

Analizando esta tabla podremos comprobar que el usuario iarmendariz, desde la misma máquina P-ARMENDARIZ, modifico el precio del producto Tejas de 20€ a 30€ el día 14/11/2015 a las 20:07:07.

SQL	Buscar	Insertar	Exportar	Importar	Privilegios	Operaciones	Disparadores	
9	root@localhost	2015-11-14 17:47:48	Cemento	50	102233	Cemento	50	1022340
10	root@localhost	2015-11-14 17:58:47	Tejas	20	40	Tejas	20	99940
11	root@localhost	2015-11-14 18:11:23	Ladrillos	13	20	Ladrillos	13	201
12	iarmendariz@P-ARMENDARIZ	2015-11-14 18:24:17	Ladrillos	13	201	Ladrillos	13	2018
13	root@localhost	2015-11-14 18:39:52	Ladrillos	13	2018	Ladrillos	13	20
14	root@localhost	2015-11-14 18:40:00	Tejas	20	99940	Tejas	20	40
15	root@localhost	2015-11-14 18:40:06	Cemento	50	1022340	Cemento	50	100
16	root@localhost	2015-11-14 18:45:38	Ladrillos	13	20	Ladrillos	17	20
17	root@localhost	2015-11-14 19:35:03	Ladrillos	17	20	Ladrillos	17	20
18	iarmendariz@P-ARMENDARIZ	2015-11-14 20:07:07	Tejas	20	40	Tejas	30	40

Figura 63. Auditoría ataque 2 MySQL

3.7.3. Análisis ataque 3

3.7.3.1. Análisis sobre SGBD sin auditoría

El análisis se realizara sobre los logs generados por el propio gestor. En el escenario que nos ocupa, necesitamos saber quién y cuándo realizo un ataque de SQL Injection, para lo cual buscaremos y analizaremos el registro “General query log”. En este log se registran las conexiones establecidas con el servidor y las declaraciones o sentencias realizadas sobre la base de datos.

Una vez localizado el log en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se produce el SQL Injection.

```
1 )                23 Prepare    SELECT `IdProducto`,`Producto`,`Precio`,`Stock`  FROM `productos`  WHERE `IdProducto` = ?
151114 22:01:44    23 Query      SELECT `productos`.`IdProducto` FROM `productos` WHERE ((`IdProducto` = 999 ) OR NOT((-1 = 0 ) ) )
BLE NAME = 'auditoria_productos'                AND IS_UPDATABLE = 'YES'            24 Query      SHOW INDEXES FROM `unir`.`auditoria_
```

Figura 64. Log ataque 3 MySQL

En la imagen anterior se puede observar que se produjo una alteración de la sentencia SQL el 14/11/2015 a las 22:01:44

En este caso nos resultará imposible conocer el autor de los hechos, puesto que la alteración se ha realizado desde el exterior aprovechándose de una vulnerabilidad de la aplicación. Y quien aparece como usuario es el usuario con que la aplicación accede a la base de datos

```
AS `checksum` ,                                    CREATE_OPTIO
22 Connect    iarmendariz@P-ARMENDARIZ on unir
```

Figura 65. Log ataque 3 MySQL

3.6.3.2. Análisis sobre SGBD con auditoría

Para este segundo modo en el que existe una auditoría de la base de datos, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

Pero en este caso, la auditoría únicamente registra las transacciones en las que existe algún tipo de modificación sobre los datos. Y en el caso de este ataque, no se ha producido ninguna alteración de datos, simplemente se ha tenido acceso a ellos, se ha podido producir un robo de datos.

3.7.4. Análisis ataque 4

3.7.4.1. Análisis sobre SGBD sin auditoría

El análisis se realizara sobre los logs generados por el propio gestor. En el escenario que nos ocupa, necesitamos saber quién y cuándo realizo un ataque de de fuerza bruta, para lo cual buscaremos y analizaremos el registro “General query log”. En este log se registran las conexiones establecidas con el servidor y las declaraciones o sentencias realizadas sobre la base de datos.

Una vez localizado el log en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se produce el ataque de fuerza bruta.

En las siguientes dos imágenes podemos observar cómo se intenta acceder a MySQL con el usuario root y probando múltiples contraseñas. El ataque se produce el 14/11/2015 hacia las 23:32 desde la maquina 192.168.2.11

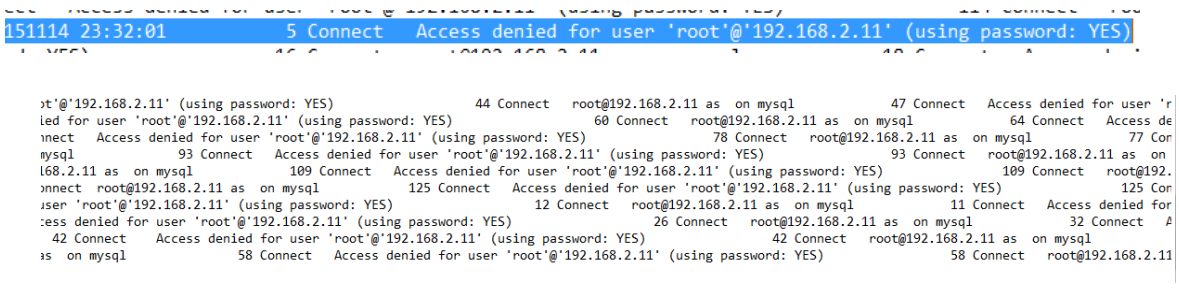


Figura 66. Log Ataque 4 MySQL

3.6.4.2. Análisis sobre SGBD con auditoría

Para este segundo modo en el que existe una auditoría de la base de datos, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

Pero en este caso, la auditoría únicamente registra las transacciones en las que existe algún tipo de modificación sobre los datos. Y en el caso de este ataque, no se ha producido ninguna alteración de datos, simplemente se ha intentado acceder a la base de datos con un ataque de fuerza bruta para descubrir la contraseña de root.

Si esta contraseña es descubierta y se accede a la base de datos para realizar cualquier tipo de modificación, entonces si se registran estos datos en la auditoría.

3.7.5. Resultados

El objetivo del análisis sobre los ataques sufridos era intentar responder a cuatro preguntas:

1. ¿Que se ha alterado?
2. ¿Cómo se ha alterado?
3. ¿Quién o desde donde lo ha alterado?
4. ¿Cuándo lo ha alterado?

En la siguiente tabla se muestra si se han alcanzado los objetivos de descubrir los ataques sufridos con los diferentes análisis.

Tipo de Análisis		Ataque 1	Ataque 2	Ataque 3	Ataque 4
Log	¿Qué?	✓	✓	✓	✓
	¿Cómo?	✓	✓	✓	✓
	¿Quién?	✓	✓	✗	✓
	¿Cuándo?	✓	✓	✓	✓
Auditoría (mediante Trigger)	¿Qué?	✓	✓	✗	✗
	¿Cómo?	✓	✓	✗	✗
	¿Quién?	✓	✓	✗	✗
	¿Cuándo?	✓	✓	✗	✗

Tabla 6. Resultados MySQL

3.8. Análisis PostgreSQL

Como se ha comentado anteriormente, el análisis se realizara sobre dos modos diferentes del gestor de base de datos, por un lado cuando únicamente se registran logs y por otro cuando además existe establecida una auditoría sobre la base de datos.

3.8.1. Análisis ataque 1

3.8.1.1. Análisis sobre SGBD sin auditoría

En este caso el análisis se tiene que realizar sobre los logs generados por el propio gestor. Dependiendo de la instalación, estos logs podrán situarse en diferentes lugares del disco. . Generalmente se sitúa en:

C:\Program Files\PostgreSQL\9.4\data\pg_log

En el escenario que nos ocupa, necesitamos saber quién y cuándo modifico el precio de un producto, para lo cual buscaremos y analizaremos el registro que genera PostgreSQL. En este log se registran las conexiones establecidas con el servidor y las declaraciones o sentencias realizadas sobre la base de datos

Una vez localizado el log en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se modifica el precio al producto.

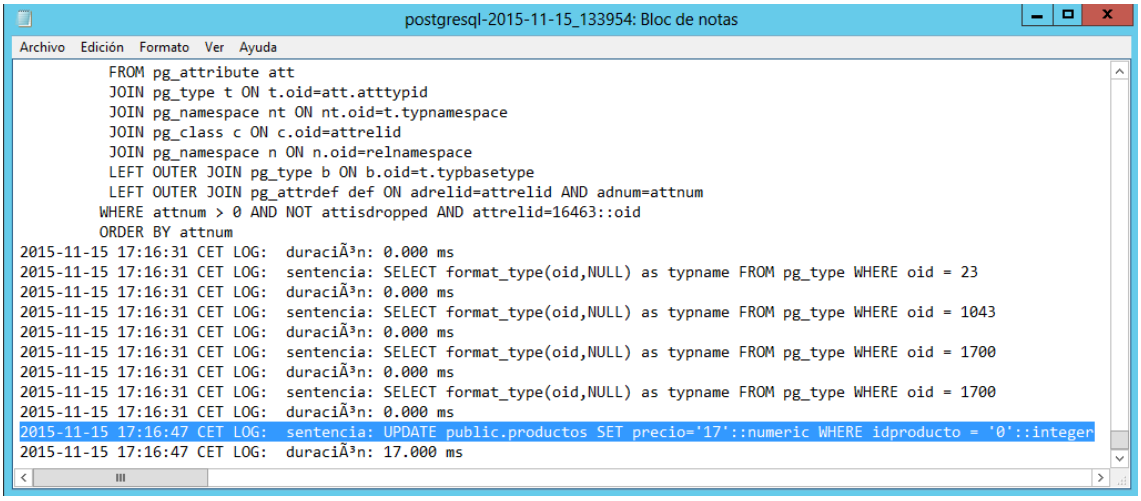


Figura 67. Log ataque 1 PostgreSQL

En la imagen anterior se puede observar que la modificación del precio del producto con IdProducto=0, que corresponde al producto “Ladrillo”, se modifico a 17€ el día 15/11/2015 a las 17:16:47.

Lo siguiente será averiguar quién realizo esa modificación, y analizando ese mismo log podremos ver que lo realizo el usuario postgres desde el mismo servidor que contiene la base de datos, a la cual se conecto el día 15/11/2015 a las 17:12:39.

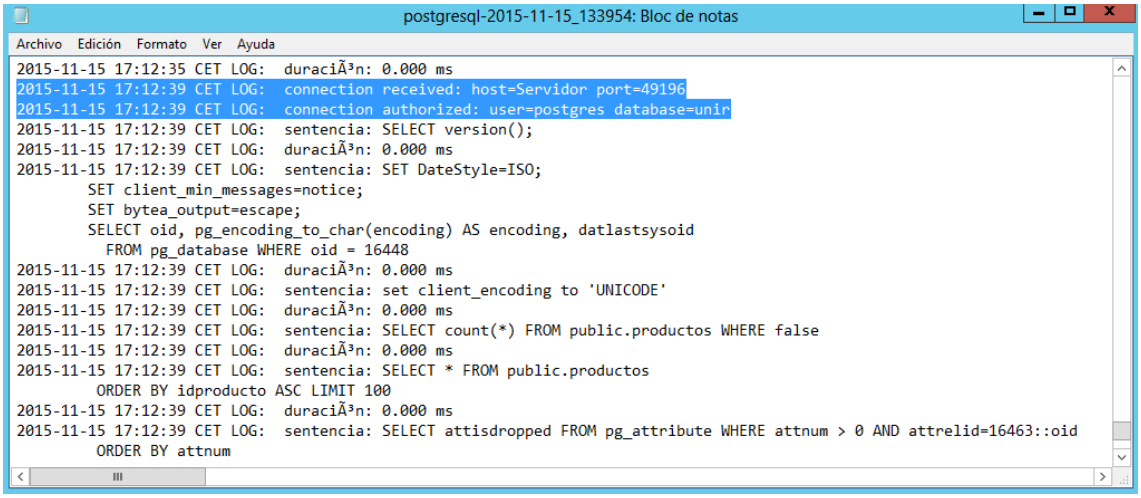


Figura 68. Log ataque 1 PostgreSQL

3.7.1.2. Análisis sobre SGBD con auditoría

Para este segundo modo en el que existe una auditoría de la base de datos, y una vez que hemos recopilado toda la información sobre cómo está establecida esa auditoría por parte de los administradores de sistemas de la empresa, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

En este caso, todas las transacciones realizadas sobre la tabla productos se auditan sobre la tabla auditoria_productos.

Analizando esta tabla podremos comprobar que el usuario postgres, desde la misma máquina de la base de datos (1/128), modifico el precio del producto Ladrillo de 13€ a 17€ el día 15/11/2015 a las 17:16:47.

1	productos	UPDATE	(2,Cemento,50,100)	(2,Cemento,50,1)	2015-11-15 15:22:39.418	postgres	192.168.2.4/32
2	productos	UPDATE	(2,Cemento,50,1)	(2,Cemento,50,11)	2015-11-15 15:22:41.387	postgres	192.168.2.4/32
3	productos	UPDATE	(2,Cemento,50,11)	(2,Cemento,50,111)	2015-11-15 16:56:24.192	postgres	192.168.2.4/32
4	productos	UPDATE	(2,Cemento,50,111)	(2,Cemento,50,100)	2015-11-15 17:04:58.209	postgres	192.168.2.4/32
9	productos	UPDATE	(0,Ladrillos,13,20)	(0,Ladrillos,17,20)	2015-11-15 17:16:47.781	postgres	::1/128

Figura 69. Auditoría ataque 1 PostgreSQL

3.8.2. Análisis ataque 2

3.8.2.1. Análisis sobre SGBD sin auditoría

En este caso el análisis se tiene que realizar sobre los logs generados por el propio gestor.

En el escenario que nos ocupa, necesitamos saber quién y cuándo modifico el precio de un producto, para lo cual buscaremos y analizaremos el registro que genera PostgreSQL. En este log se registran las conexiones establecidas con el servidor y las declaraciones o sentencias realizadas sobre la base de datos

Una vez localizado el log en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se modifica el precio al producto.

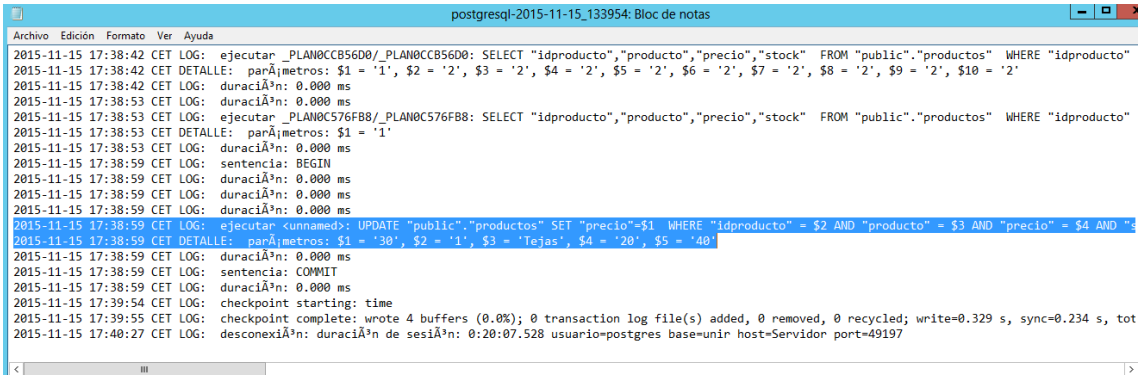


Figura 70. Log Ataque 2 PostgreSQL

```
2015-11-15 17:38:59 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 17:38:59 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 17:38:59 CET LOG:  ejecutar <unnamed>: UPDATE "public"."productos" SET "precio"=$1  WHERE "idproducto" = $2
2015-11-15 17:38:59 CET DETALLE:  parÃ¡metros: $1 = '30', $2 = '1', $3 = 'Tejas', $4 = '20', $5 = '40'
```

Figura 71. Log Ataque 2 PostgreSQL

En las dos imágenes anteriores se puede observar que la modificación del precio del producto con IdProducto=1, que corresponde al producto “Tejas”, se modifico a 30€ el día 15/11/2015 a las 17:38:59.

Lo siguiente será averiguar quién realizo esa modificación, y analizando el log podemos ver que lo realizo el usuario iarmendariz desde la maquina P-ARMENDARIZ, a la cual se conecto el día 15/11/2015 a las 17:38:39.

```
2015-11-15 17:27:12 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 17:27:12 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 17:27:12 CET LOG:  ejecutar _PLAN0CCB56D0/_PLAN0CCB56D0: SELECT "idproducto","producto"
2015-11-15 17:27:12 CET DETALLE:  parÃ¡metros: $1 = '0', $2 = '1', $3 = '2', $4 = '2', $5 = '2', $
2015-11-15 17:27:12 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 17:37:13 CET LOG:  sentencia: DEALLOCATE "_PLAN0CCB56D0"
2015-11-15 17:37:13 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 17:37:13 CET LOG:  desconexiÃ³n: duraciÃ³n de sesiÃ³n: 0:10:00.131 usuario=iarmendariz
2015-11-15 17:38:39 CET LOG:  connection received: host=P-ARMENDARIZ port=50275
2015-11-15 17:38:39 CET LOG:  connection authorized: user=iarmendariz database=unir
2015-11-15 17:38:39 CET LOG:  sentencia: select oid, typbasetype from pg_type where typname = 'lo'
2015-11-15 17:38:39 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 17:38:39 CET LOG:  sentencia: set client_encoding to 'WIN1252'
2015-11-15 17:38:39 CET LOG:  duraciÃ³n: 16.000 ms
```

Figura 72. Log Ataque 2 PostgreSQL

3.7.2.2. Análisis sobre SGBD con auditoría

Para este segundo modo en el que existe una auditoría de la base de datos, y una vez que hemos recopilado toda la información sobre cómo está establecida esa auditoría por parte de los administradores de sistemas de la empresa, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

En este caso, todas las transacciones realizadas sobre la tabla productos se auditan sobre la tabla auditoria_productos.

Analizando esta tabla podremos comprobar que el usuario iarmendariz, desde la máquina 192.168.2.4, modifico el precio del producto Tejas de 20€ a 30€ el día 15/11/2015 a las 17:38:59.

idproducto [PK] serial	NombreTabla character(45)	Operacion character(10)	ValorViejo text	ValorNuevo text	Fecha timestamp without time zone	Usuario character(45)	Host character(45)
1	productos	UPDATE	(2,Cemento,50,100)	(2,Cemento,50,1)	2015-11-15 15:22:39.418	postgres	192.168.2.4/32
2	productos	UPDATE	(2,Cemento,50,1)	(2,Cemento,50,11)	2015-11-15 15:22:41.387	postgres	192.168.2.4/32
3	productos	UPDATE	(2,Cemento,50,11)	(2,Cemento,50,111)	2015-11-15 16:56:24.192	postgres	192.168.2.4/32
4	productos	UPDATE	(2,Cemento,50,111)	(2,Cemento,50,100)	2015-11-15 17:04:58.209	postgres	192.168.2.4/32
9	productos	UPDATE	(0,Ladrillos,13,20)	(0,Ladrillos,17,20)	2015-11-15 17:16:47.781	postgres	::1/128
10	productos	UPDATE	(1,Tejas,20,40)	(1,Tejas,30,40)	2015-11-15 17:38:59.157	iarmendariz	192.168.2.4/32

Figura 73. Auditoría ataque 2 PostgreSQL

3.8.3. Análisis ataque 3

3.8.3.1. Análisis sobre SGBD sin auditoría

El análisis se realizara sobre los logs generados por el propio gestor. En el escenario que nos ocupa, necesitamos saber quién y cuándo realizo un ataque de SQL Injection, para lo cual buscaremos y analizaremos el log sobre el que se registran las conexiones

establecidas con el servidor y las declaraciones o sentencias realizadas sobre la base de datos.

Una vez localizado el log en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se produce el SQL Injection.

```
2015-11-15 18:15:23 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 18:15:23 CET LOG:  sentencia: DEALLOCATE "_PLAN0C576FB8"
2015-11-15 18:15:23 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 18:15:40 CET LOG:  sentencia: SELECT "public"."productos"."idproducto" FROM "public"."productos" WHERE (("idproducto" = 999 ) OR NOT((-1 = 0 ) ) )
2015-11-15 18:15:40 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 18:15:40 CET LOG:  duraciÃ³n: 0.000 ms
2015-11-15 18:15:40 CET LOG:  duraciÃ³n: 0.000 ms
```

Figura 74. Log ataque 3 PostgreSQL

En la imagen anterior se puede observar que se produjo una alteración de la sentencia SQL el 15/11/2015 a las 18:15:40

En este caso nos resultará imposible conocer el autor de los hechos, puesto que la alteración se ha realizado desde el exterior aprovechándose de una vulnerabilidad de la aplicación. Y quien aparece como usuario es el usuario con que la aplicación accede a la base de datos

```
2015-11-15 18:13:54 CET LOG:  desconexiÃ³n: duraciÃ³n de sesiÃ³n: 0:10:09.710 usuario=postgres base=unir
2015-11-15 18:15:06 CET LOG:  connection received: host=P-ARMENDARIZ port=50285
2015-11-15 18:15:06 CET LOG:  connection authorized: user=postgres database=unir
2015-11-15 18:15:06 CET LOG:  sentencia: select oid, typbasetype from pg_type where typname = 'lo'
```

Figura 75. Log ataque 3 PostgreSQL

3.8.3.2. Análisis sobre SGBD con auditoría

Para este segundo modo en el que existe una auditoría de la base de datos, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

Pero en este caso, la auditoría únicamente registra las transacciones en las que existe algún tipo de modificación sobre los datos. Y en el caso de este ataque, no se ha producido ninguna alteración de datos, simplemente se ha tenido acceso a ellos, se ha podido producir un robo de datos

3.8.4. Análisis ataque 4

3.8.4.1. Análisis sobre SGBD sin auditoría

El análisis se realizara sobre los logs generados por el propio gestor. En el escenario que nos ocupa, necesitamos saber quién y cuándo realizo un ataque de de fuerza bruta, para lo cual buscaremos y analizaremos el log en el que se registran las conexiones establecidas con el servidor y las declaraciones o sentencias realizadas sobre la base de datos.

Una vez localizado el log en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se produce el ataque de fuerza bruta.

En la siguiente imagen podemos observar cómo se intenta acceder a PostgreSQL con el usuario postgres y probando múltiples contraseñas. El ataque se produce el 15/11/2015 de las 18:34 en adelante desde la maquina 192.168.2.11.

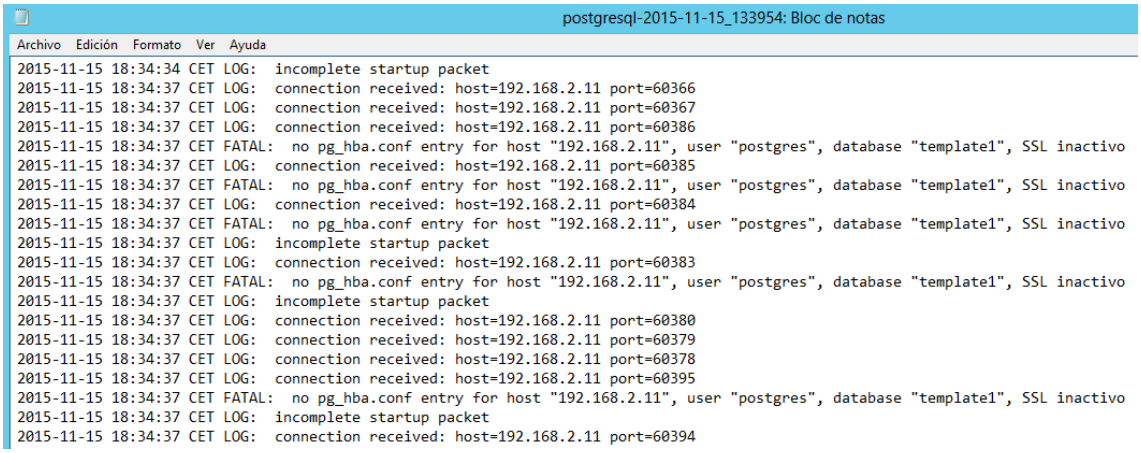


Figura 76. Log ataque 4 PostgreSQL

3.8.4.2. Análisis sobre SGBD con auditoría

Para este segundo modo en el que existe una auditoría de la base de datos, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

Pero en este caso, la auditoría únicamente registra las transacciones en las que existe algún tipo de modificación sobre los datos. Y en el caso de este ataque, no se ha producido ninguna alteración de datos, simplemente se ha intentado acceder a la base de datos con un ataque de fuerza bruta para descubrir la contraseña de root.

Si esta contraseña es descubierta y se accede a la base de datos para realizar cualquier tipo de modificación, entonces si se registran estos datos en la auditoría.

3.8.5. Resultados

El objetivo del análisis sobre los ataques sufridos era intentar responder a cuatro preguntas:

5. ¿Que se ha alterado?
6. ¿Cómo se ha alterado?
7. ¿Quién o desde donde lo ha alterado?
8. ¿Cuándo lo ha alterado?

En la siguiente tabla se muestra si se han alcanzado los objetivos de descubrir los ataques sufridos con los diferentes análisis.

Tipo de Análisis		Ataque 1	Ataque 2	Ataque 3	Ataque 4
Log	¿Qué?	✓	✓	✓	✓
	¿Cómo?	✓	✓	✓	✓
	¿Quién?	✓	✓	✗	✓
	¿Cuándo?	✓	✓	✓	✓
Auditoría (mediante Trigger)	¿Qué?	✓	✓	✗	✗
	¿Cómo?	✓	✓	✗	✗
	¿Quién?	✓	✓	✗	✗
	¿Cuándo?	✓	✓	✗	✗

Tabla 7. Resultados PorstgreSQL

3.9. Análisis Microsoft SQL Server

Como se ha comentado anteriormente, el análisis se realizara sobre dos modos diferentes del gestor de base de datos, por un lado cuando únicamente se registran logs y por otro cuando además existe establecida una auditoría sobre la base de datos.

3.9.1. Análisis ataque 1

3.9.1.1. Análisis sobre SGBD sin auditoría

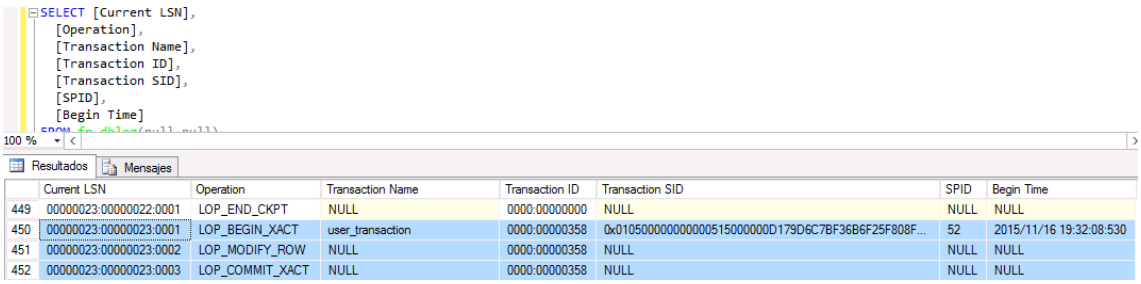
Como se ha comentado anteriormente, todas las bases de datos de SQL Server tienen un registro de transacciones que registra todas las transacciones y las modificaciones que cada transacción realiza en la base de datos.

El archivo con este registro se guarda en formato LDF, pero si no se utiliza alguna herramienta especial para visualizarlo muestra entradas ininteligibles, así que estas entradas no pueden ser leídas directamente.

Existen dos maneras de visualizar el log de transacciones sin utilizar ninguna herramienta de terceros.

1. Usando la función fn_dblog:

Podemos utilizar la función no documentada fn_dblog para desgranar el contenido del registro de transacciones:



	Current LSN	Operation	Transaction Name	Transaction ID	Transaction SID	SPID	Begin Time
449	00000023:00000022:0001	LOP_END_CKPT	NULL	0000:00000000	NULL	NULL	NULL
450	00000023:00000023:0001	LOP_BEGIN_XACT	user_transaction	0000:00000358	0x0105000000000005150000000D179D6C7BF36B6F25F808F...	52	2015/11/16 19:32:08:530
451	00000023:00000023:0002	LOP_MODIFY_ROW	NULL	0000:00000358	NULL	NULL	NULL
452	00000023:00000023:0003	LOP_COMMIT_XACT	NULL	0000:00000358	NULL	NULL	NULL

Figura 77. Función fn_dblog ataque 1 MS SQL Server

En el caso que nos ocupa, la modificación del campo precio, la operación se ha registrado en tres filas.

Las operaciones LOP_BEGIN_XACT y LOP_COMMIT_XACT indican el comienzo y fin de la transacción. La operación LOP_MODIFY_ROW indica que se ha producido una modificación en un registro

Se puede apreciar que la transacción comenzó el 16/11/2015 a las 19:32:08. Pero interpretar el resto de la información es una tarea muy compleja porque esa función devuelve 129 columnas ilegibles que habrá que traducir y no hay documentación oficial disponible para esta función.

2. Usando el comando DBCC LOG:

Mediante este comando se puede leer el contenido de los archivos de transacciones LDF. Pero como ocurre con el anterior, no existe documentación oficial sobre este comando.

Este comando admite como parámetros el nombre de la base de datos y un código numérico que indica el tipo de información que se desea obtener del log:

0- Información Mínima (Current LSN, Operation, Context, Transaction ID)

450	00000023:00000023:0001	LOP_BEGIN_XACT	LCX_NULL	0000:00000358	0
451	00000023:00000023:0002	LOP_MODIFY_ROW	LCX_CLUSTERED	0000:00000358	0
452	00000023:00000023:0003	LOP_COMMIT_XACT	LCX_NULL	0000:00000358	0

Figura 78. Comando DBCC LOG ataque 1 MS SQL Server

1- Algo más de información

2- Bastante más información

3- Toda la información

00000023:00000023:0001	LOP_BEGIN_XACT	LCX_NULL	0000:00000358	0	0x0000	76	144	00000000:00000000:0000	0x0002	9170	NULL	NULL
00000023:00000023:0002	LOP_MODIFY_ROW	LCX_CLUSTERED	0000:00000358	0	0x0000	62	120	00000023:00000023:0001	0x0002	217	7205759...	dbo.productos.PK_...
00000023:00000023:0003	LOP_COMMIT_XACT	LCX_NULL	0000:00000358	0	0x0000	80	84	00000023:00000023:0001	0x0002	90	NULL	NULL

Figura 79. Comando DBCC LOG ataque 1 MS SQL Server

4- Volcado en hexadecimal de cierta información

Como se ha podido comprobar, resulta complicado poder interpretar la información del LOG utilizando la función fn_dblog y el comando DBCC LOG. Es por ello que voy a utilizar una herramienta de terceros para poder analizar los transaction logs.

Para este caso, voy a utilizar ApexSQL Log. Con esta herramienta es sencillo analizar los logs de SQL server. Como se puede apreciar en la imagen siguiente, el usuario administrador realizo una modificación en la tabla productos el 16/11/2015 a las 19:32:08

UPDATE	dbo	productos	SERVIDOR\Administrador	2015-11-16 19:32:08	2015-11-16 19:32:08	0000:00000358	00000023:00000023:0002

Figura 80. ApexSQL Log ataque 1 MS SQL Server

Y viendo el detalle del registro podemos apreciar que se cambio el precio del producto Ladrillos de 13 a 17

Field	Type	Old Value	New Value
IdProducto	int	0	0
Producto	nvarchar(50)	Ladrillos	Ladrillos
Precio	int	13	17
Stock	int	20	20

Figura 81. Detalle ApexSQL ataque 1 Log MS SQL Server

3.9.1.2. Análisis sobre SGBD con auditoría

Se han establecido tres métodos para auditar eventos: auditoría, triggers y SQL Profiler. Los resultados sobre el ataque sufrido con los distintos métodos son los siguientes:

a) Mediante Auditoría

Analizando la auditoría podemos comprobar que el usuario administrador modifico el precio del producto Ladrillo de 13€ a 17€ el día 18/11/2015 a las 17:39:35.

Fecha	Hora del evento	Nombre de instancia del servidor	Id. de acción	Tipo de clase	Número de
18/11/2015 17:39:35	17:39:35.8481546	SERVIDOR	SELECT	TABLE	1
18/11/2015 17:39:35	17:39:35.8481546	SERVIDOR	UPDATE	TABLE	1

Detalles de las filas seleccionadas:

Fecha 18/11/2015 17:39:35
Registro Recopilación de auditoría (Auditoria_productos)

Hora del evento 17:39:35.8481546
Nombre de instancia del servidor SERVIDOR
Id. de acción UPDATE
Tipo de clase TABLE
Número de secuencias 1
Correcto True
Máscara de bits de permiso 0x0000000000000002
Permiso de columna True
Id. de sesión 59
Id. de la entidad de seguridad del servidor 260
Id. de la entidad de seguridad de la base de datos 1
Id. de la entidad de seguridad del servidor de destino 0
Id. de la entidad de seguridad de la base de datos de destino 0
Id. de objeto 885578193
Nombre de la entidad de seguridad del servidor de la sesión SERVIDOR\Administrador
Nombre de la entidad de seguridad del servidor SERVIDOR\Administrador
SID de la entidad de seguridad del servidor 0x150000052100020912121419919154182242951281432332441 00
Nombre de la entidad de seguridad de la base de datos dbo
Nombre de la entidad de seguridad del servidor de destino
SID de la entidad de seguridad del servidor de destino NULL
Nombre de la entidad de seguridad de la base de datos de destino
Nombre de la base de datos Unir
Nombre del esquema dbo
Nombre de objeto productos
Instrucción update productos set precio=17 where IdProducto=0
Información adicional
Nombre de archivo C:\auditoria\Auditoria_productos_FB2ECA62-070F-4848-8105-4E7B068DCB66_0_130923409719080000.sqlaudit
Desplazamiento de archivo 14336
Id. de evento definido por el usuario 0

Figura 82. Auditoría ataque 1 MS SQL Server

b) Mediante Triggers

Analizando los registros generados por el disparador en la tabla auxiliar podemos comprobar que el usuario administrador, desde el equipo SERVIDOR, modifico el precio del producto Ladrillo de 13€ a 17€ el día 18/11/2015 a las 18:39:35.

8	D	productos	<IdProducto=12>	Producto	pala	NULL	2015-11-18 18:35:52.717	SERVIDOR\Administrador
9	D	productos	<IdProducto=12>	Precio	233	NULL	2015-11-18 18:35:52.717	SERVIDOR\Administrador
10	D	productos	<IdProducto=12>	Stock	33	NULL	2015-11-18 18:35:52.717	SERVIDOR\Administrador
11	U	productos	<IdProducto=0>	Precio	13	17	2015-11-18 18:39:35.847	SERVIDOR\Administrador

Figura 83. Trigger ataque 1 MS SQL Server

c) Mediante SQL Server Profiler

Visualizando los rastros de los eventos en SQL Server Profiler encontramos que el usuario administrador, desde el equipo SERVIDOR, modifico el precio del producto Ladrillo de 13€ a 17€ el día 18/11/2015 a las 18:39:35.

SQL:BatchStarting	update productos set precio=17 where IdProducto=0	Microsoft S...	Administrador	SERVIDOR\Administrador
SQL:BatchCompleted	update productos set precio=17 where IdProducto=0	Microsoft S...	Administrador	SERVIDOR\Administrador

Figura 84. Server Profiler ataque 1 MS SQL Server

SERVIDOR\Administrador					2124	59	2015-11-18 18:39:35.847	
SERVIDOR\Administrador	15	479	11	52	2124	59	2015-11-18 18:39:35.847	2015-11-18 18:39:35.900

Figura 85. Server Profiler ataque 1 MS SQL Server

3.9.2. Análisis ataque 2

3.9.2.1. Análisis sobre SGBD sin auditoría

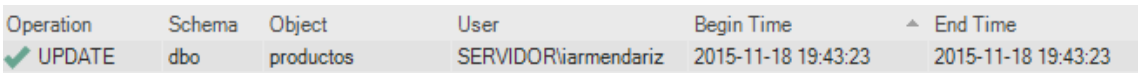
Como se ha comentado anteriormente, todas las bases de datos de SQL Server tienen un registro de transacciones que registra todas las transacciones y las modificaciones que cada transacción realiza en la base de datos.

El archivo con este registro se guarda en formato LDF, pero si no se utiliza alguna herramienta especial para visualizarlo muestra entradas ininteligibles, así que estas entradas no pueden ser leídas directamente.

Existen dos maneras de visualizar el log de transacciones sin utilizar ninguna herramienta de terceros, utilizando la función fn_dblog o el comando DBCC LOG, pero como se ha

podido apreciar en el ataque anterior, los datos son prácticamente ilegibles y se necesitaría mucho tiempo para poder interpretarlos. Debido a ello vamos a utilizar una aplicación de terceros.

Para este caso, voy a utilizar ApexSQL Log. Con esta herramienta es sencillo analizar los logs de SQL server. Como se puede apreciar en la imagen siguiente, el usuario iarmendariz realizo una modificación en la tabla productos el 18/11/2015 a las 19:43:23



Operation	Schema	Object	User	Begin Time	End Time
UPDATE	dbo	productos	SERVIDOR\iarmendariz	2015-11-18 19:43:23	2015-11-18 19:43:23

Figura 86. ApexSQL Log ataque 2 MS SQL Server

Y viendo el detalle del registro podemos apreciar que se cambio el precio del producto Tejas de 20 a 30

Field	Type	Old Value	New Value
IdProducto	int	1 ...	1
Producto	nvarchar(50)	Tejas ...	Tejas
Precio	int	20 ...	**30**
Stock	int	40 ...	40

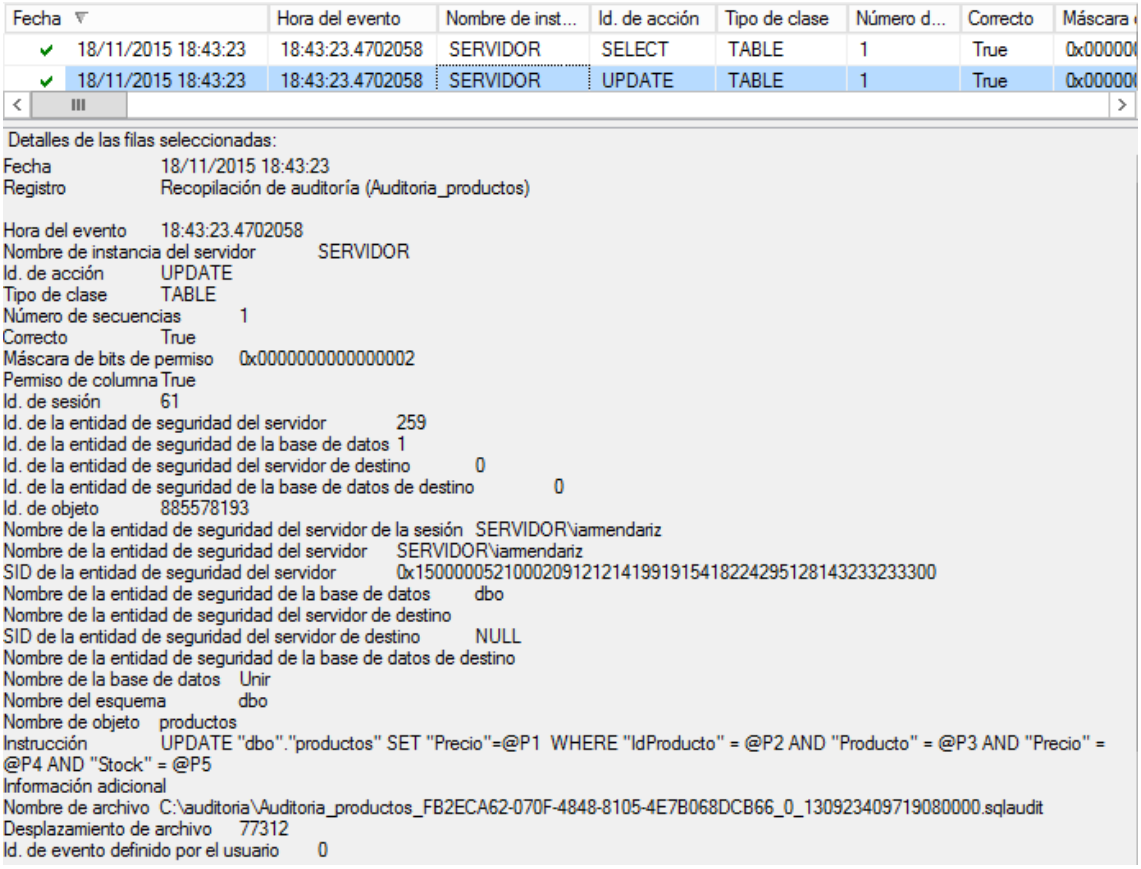
Figura 87. Detalle ApexSQL Log ataque 2 MS SQL Server

3.9.2.2. Análisis sobre SGBD con auditoría

Se han establecido tres métodos para auditar eventos: auditoría, triggers y SQL Profiler. Los resultados sobre el ataque sufrido con los distintos métodos son los siguientes:

a) Mediante Auditoría

Analizando la auditoría podemos comprobar que el usuario iarmendariz modifico el precio del producto Tejas el día 18/11/2015 a las 19:43:23 (el registro de auditoría marca una hora menos).



Fecha	Hora del evento	Nombre de inst...	Id. de acción	Tipo de clase	Número d...	Correcto	Máscara
18/11/2015 18:43:23	18:43:23.4702058	SERVIDOR	SELECT	TABLE	1	True	0x00000
18/11/2015 18:43:23	18:43:23.4702058	SERVIDOR	UPDATE	TABLE	1	True	0x00000

Detalles de las filas seleccionadas:

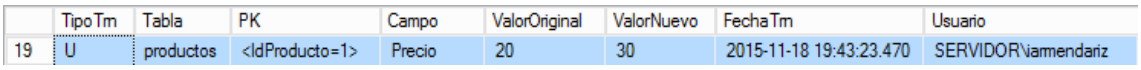
Fecha 18/11/2015 18:43:23
Registro Recopilación de auditoría (Auditoria_productos)

Hora del evento 18:43:23.4702058
Nombre de instancia del servidor SERVIDOR
Id. de acción UPDATE
Tipo de clase TABLE
Número de secuencias 1
Correcto True
Máscara de bits de permiso 0x0000000000000002
Permiso de columna True
Id. de sesión 61
Id. de la entidad de seguridad del servidor 259
Id. de la entidad de seguridad de la base de datos 1
Id. de la entidad de seguridad del servidor de destino 0
Id. de la entidad de seguridad de la base de datos de destino 0
Id. de objeto 885578193
Nombre de la entidad de seguridad del servidor de la sesión SERVIDOR\iarmendariz
Nombre de la entidad de seguridad del servidor SERVIDOR\iarmendariz
SID de la entidad de seguridad del servidor 0x150000052100020912121419919154182242951281432332333 00
Nombre de la entidad de seguridad de la base de datos dbo
Nombre de la entidad de seguridad del servidor de destino
SID de la entidad de seguridad del servidor de destino NULL
Nombre de la entidad de seguridad de la base de datos de destino
Nombre de la base de datos Unir
Nombre del esquema dbo
Nombre de objeto productos
Instrucción UPDATE "dbo"."productos" SET "Precio"=@P1 WHERE "IdProducto" = @P2 AND "Producto" = @P3 AND "Precio" = @P4 AND "Stock" = @P5
Información adicional
Nombre de archivo C:\auditoria\Auditoria_productos_FB2ECA62-070F-4848-8105-4E7B068DCB66_0_130923409719080000.sqlaudit
Desplazamiento de archivo 77312
Id. de evento definido por el usuario 0

Figura 88. Auditoría ataque 2 MS SQL Server

b) Mediante Triggers

Analizando los registros generados por el disparador en la tabla auxiliar podemos comprobar que el usuario iarmendariz, modifico el precio del producto Tejas de 20€ a 30€ el día 18/11/2015 a las 19:43:23.



	TipoTm	Tabla	PK	Campo	ValorOriginal	ValorNuevo	FechaTm	Usuario
19	U	productos	<IdProducto=1>	Precio	20	30	2015-11-18 19:43:23.470	SERVIDOR\iarmendariz

Figura 89. Trigger ataque 2 MS SQL Server

c) Mediante SQL Server Profiler

Visualizando los rastros de los eventos en SQL Server Profiler encontramos que el usuario iarmendariz, desde el equipo SERVIDOR, modifico el precio del producto Tejas de 20€ a 30€ el día 18/11/2015 a las 19:43:23.

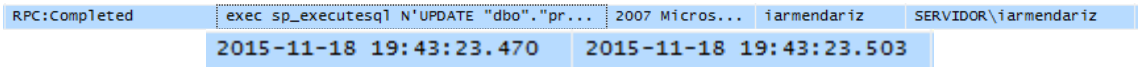


Figura 90. Server Profiler ataque 2 MS SQL Server

Figura 91. Server Profiler ataque 2 MS SQL Server

3.9.3. Análisis ataque 3

3.9.3.1. Análisis sobre SGBD sin auditoría

Como se ha comentado anteriormente, todas las bases de datos de SQL Server tienen un registro de transacciones que registra todas las transacciones y las modificaciones que cada transacción realiza en la base de datos.

Pero el registro de transacciones únicamente registra modificaciones y no consultas. En el caso que nos ocupa, el ataque de SQL Injection se ha realizado para leer información, no se ha producido ninguna alteración de datos, simplemente se ha tenido acceso a ellos, se ha podido producir un robo de datos.

Mediante el análisis del registro de transacciones no podremos obtener ningún dato de este ataque en concreto.

3.9.3.2. Análisis sobre SGBD con auditoría

Se han establecido tres métodos para auditar eventos: auditoría, triggers y SQL Profiler. Los resultados sobre el ataque sufrido con los distintos métodos son los siguientes:

a) Mediante Auditoría

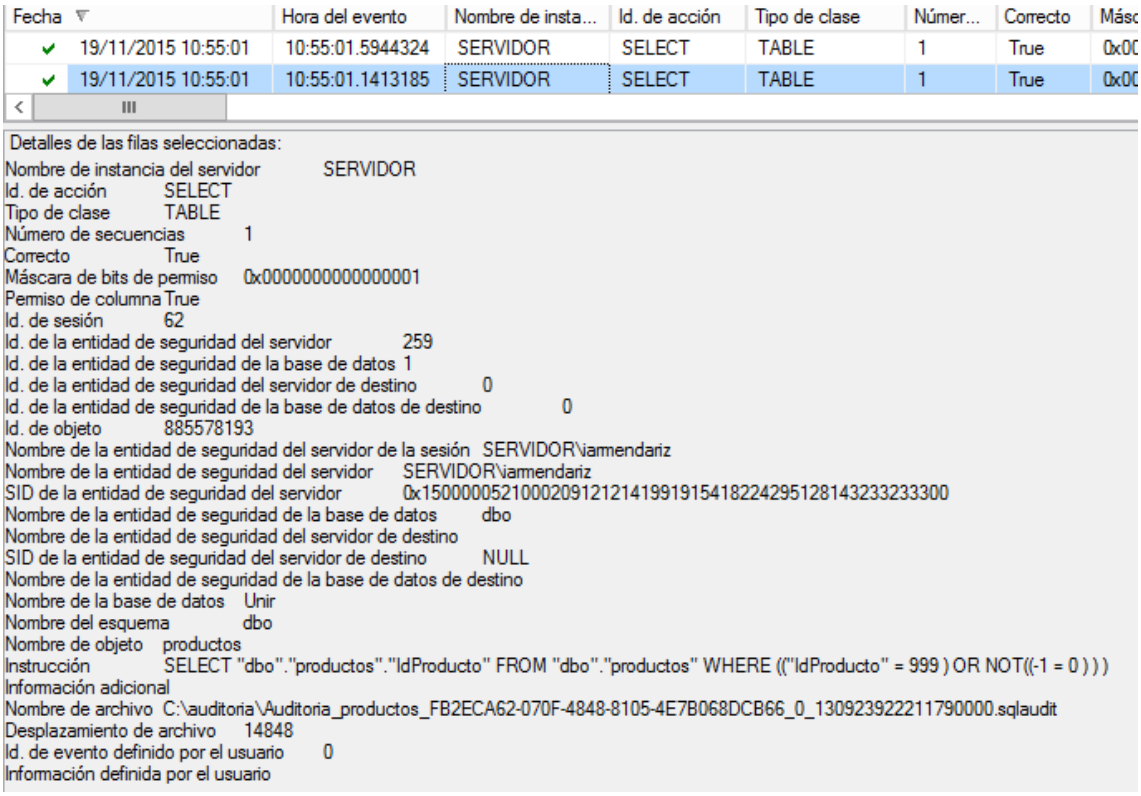
Analizando la auditoría podemos comprobar que se produjo una alteración de la sentencia SQL el día 19/11/2015 a las 11:55:01 (el registro de auditoría marca una hora menos).

Se puede comprobar también, que la auditoría traduce la expresión introducida:

SELECT "dbo"."productos"."IdProducto" FROM "dbo"."productos" WHERE "IdProducto" = 999 OR 1=1

Por:

SELECT "dbo"."productos"."IdProducto" FROM "dbo"."productos" WHERE (("IdProducto" = 999) OR NOT((-1 = 0)))



Fecha	Hora del evento	Nombre de insta...	Id. de acción	Tipo de clase	Númer...	Correcto	Másc
19/11/2015 10:55:01	10:55:01.5944324	SERVIDOR	SELECT	TABLE	1	True	0x00
19/11/2015 10:55:01	10:55:01.1413185	SERVIDOR	SELECT	TABLE	1	True	0x00

Detalles de las filas seleccionadas:

Nombre de instancia del servidor SERVIDOR
Id. de acción SELECT
Tipo de clase TABLE
Número de secuencias 1
Correcto True
Máscara de bits de permiso 0x0000000000000001
Permiso de columna True
Id. de sesión 62
Id. de la entidad de seguridad del servidor 259
Id. de la entidad de seguridad de la base de datos 1
Id. de la entidad de seguridad del servidor de destino 0
Id. de la entidad de seguridad de la base de datos de destino 0
Id. de objeto 885578193
Nombre de la entidad de seguridad del servidor de la sesión SERVIDOR\iarmendariz
Nombre de la entidad de seguridad del servidor SERVIDOR\iarmendariz
SID de la entidad de seguridad del servidor 0x150000052100020912121419919154182242951281432333333 00
Nombre de la entidad de seguridad de la base de datos dbo
Nombre de la entidad de seguridad del servidor de destino
SID de la entidad de seguridad del servidor de destino NULL
Nombre de la entidad de seguridad de la base de datos de destino
Nombre de la base de datos Unir
Nombre del esquema dbo
Nombre de objeto productos
Instrucción SELECT "dbo"."productos"."IdProducto" FROM "dbo"."productos" WHERE (("IdProducto" = 999) OR NOT((-1 = 0)))
Información adicional
Nombre de archivo C:\auditoria\Auditoria_productos_FB2ECA62-070F-4848-8105-4E7B068DCB66_0_130923922211790000.sqlaudit
Desplazamiento de archivo 14848
Id. de evento definido por el usuario 0
Información definida por el usuario

Figura 92. Auditoría ataque 3 MS SQL Server

En este caso nos resultará imposible conocer el autor de los hechos, puesto que la alteración se ha realizado desde el exterior aprovechándose de una vulnerabilidad de la aplicación. Y quien aparece como usuario es el usuario con que la aplicación accede a la base de datos.

b) Mediante Triggers

Para este segundo modo en el que existe una auditoría mediante disparadores o triggers, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

Pero en este caso, los desencadenadores únicamente registran las transacciones en las que existe algún tipo de modificación sobre los datos. Y en el caso de este ataque, no se ha producido ninguna alteración de datos, simplemente se ha tenido acceso a ellos, se ha podido producir un robo de datos.

c) Mediante SQL Server Profiler

Visualizando los rastros de los eventos en SQL Server Profiler podemos comprobar que se produjo una alteración de la sentencia SQL el día 19/11/2015 a las 11:55:01

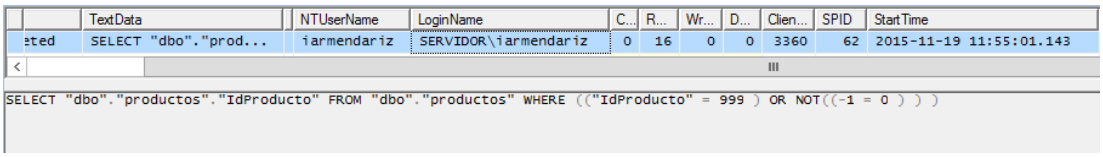


Figura 93. Server Profiler ataque 3 MS SQL Server

Nos resultará imposible conocer el autor de los hechos, puesto que la alteración se ha realizado desde el exterior aprovechándose de una vulnerabilidad de la aplicación. Y quien aparece como usuario es el usuario con que la aplicación accede a la base de datos.

3.9.4. Análisis ataque 4

3.9.4.1. Análisis sobre SGBD sin auditoría

Existen dos opciones para visualizar los intentos de acceso fallidos a SQL Server sin valernos de la auditoría. Pero para que se registren estos intentos de Login es necesario tener activado esta opción en las características de seguridad del servidor, si bien es cierto que por defecto suele venir activado.

La característica de seguridad es la siguiente:

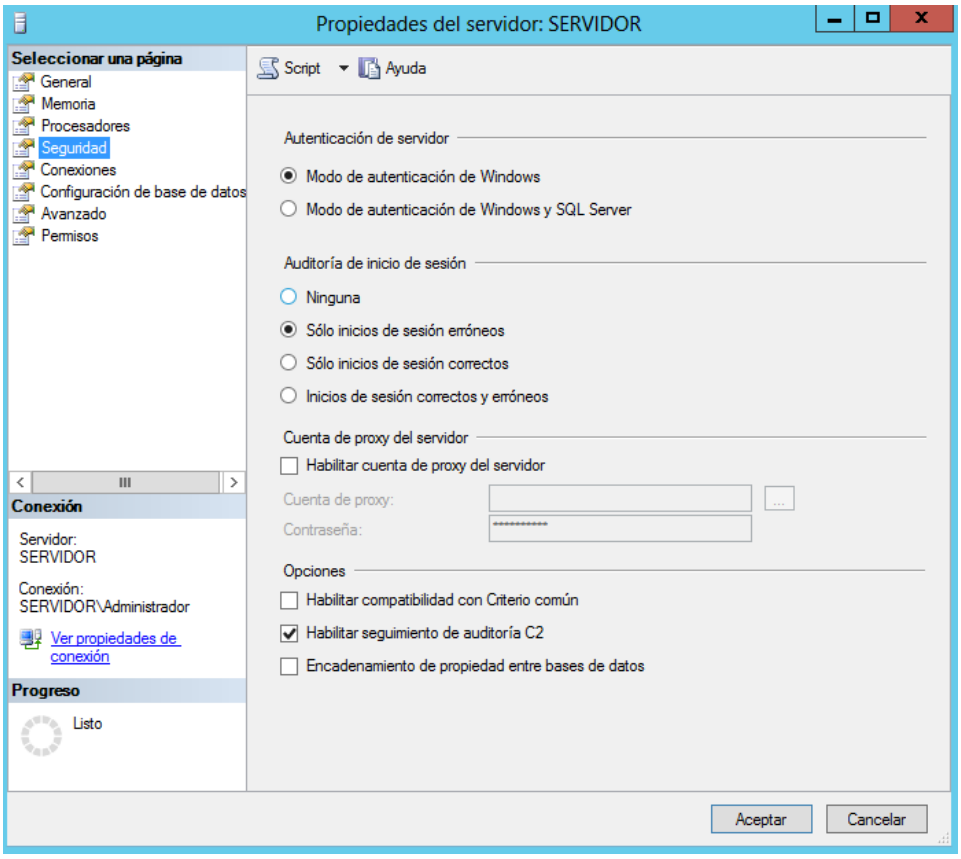


Figura 94. Propiedades servidor ataque 3 MS SQL Server

Teniendo esa característica activada esos intentos de acceso fallidos se pueden visualizar de dos maneras:

1. Visos de eventos de Windows:

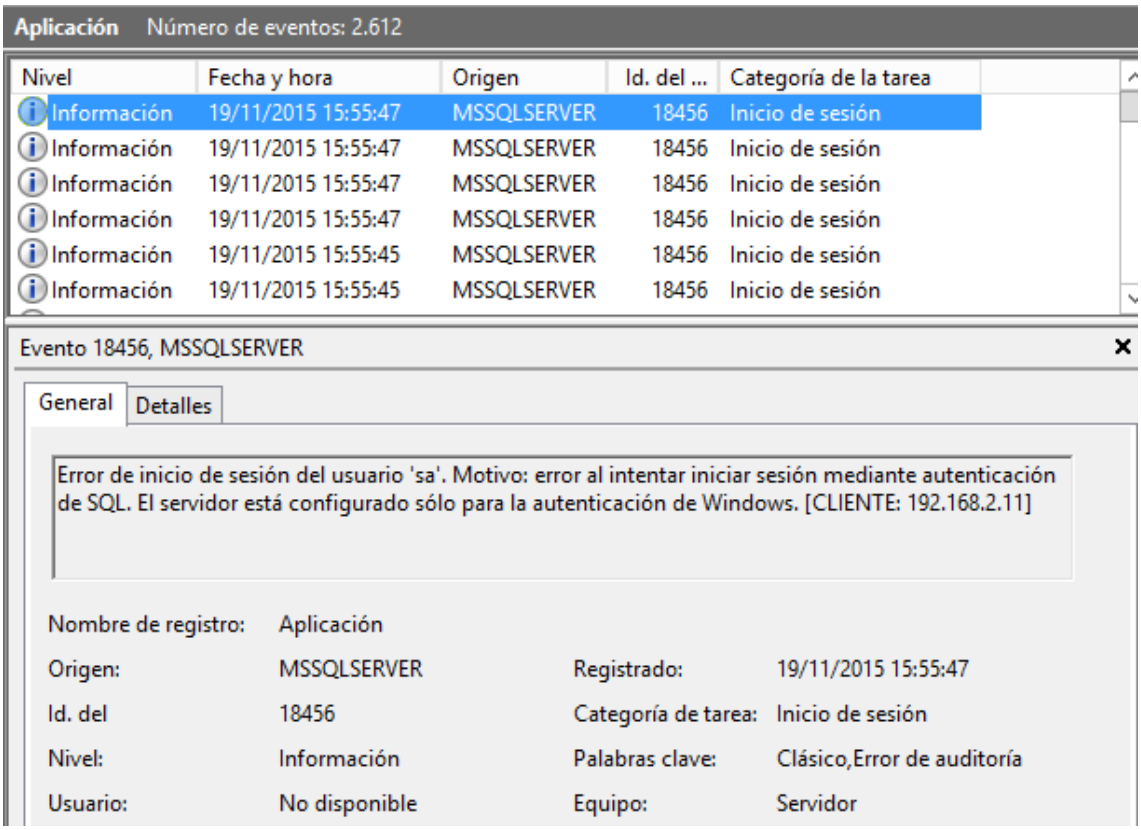


Figura 95. Visor de eventos ataque 3 MS SQL Server

Como se puede apreciar en la imagen superior, se han intentado una serie de inicios de sesión en SQL Server el 19/11/2015 desde las 15:55. El usuario utilizado para iniciar sesión ha sido “sa” (usuario administrador de SQL server) y se ha intentado este acceso desde la maquina con IP 192.168.2.11

2. Procedimiento xp_readerrorlog

Otra manera de visualizar estos intentos fallidos de sesión es utilizando el procedimiento xp_readerrorlog

SQLQuery2.sql - SE...Administrador (65))* SQLQuery1.sql - SE...Administrador (60))

```
EXEC xp_readerrorlog;
```

100 %

Resultados | Mensajes

	LogDate	ProcessInfo	Text
508	2015-11-19 15:55:47.750	Logon	Login failed for user 'sa'. Motivo: error al intentar iniciar sesión mediante autenticación de SQL. ...
509	2015-11-19 15:55:47.750	Logon	Error: 18456, gravedad: 14, estado: 58.
510	2015-11-19 15:55:47.750	Logon	Login failed for user 'sa'. Motivo: error al intentar iniciar sesión mediante autenticación de SQL. ...
511	2015-11-19 15:55:47.750	Logon	Error: 18456, gravedad: 14, estado: 58.
512	2015-11-19 15:55:47.750	Logon	Login failed for user 'sa'. Motivo: error al intentar iniciar sesión mediante autenticación de SQL. ...
513	2015-11-19 15:55:47.750	Logon	Error: 18456, gravedad: 14, estado: 58.
514	2015-11-19 15:55:47.750	Logon	Login failed for user 'sa'. Motivo: error al intentar iniciar sesión mediante autenticación de SQL. ...
515	2015-11-19 15:55:47.750	Logon	Error: 18456, gravedad: 14, estado: 58.
516	2015-11-19 15:55:47.750	Logon	Login failed for user 'sa'. Motivo: error al intentar iniciar sesión mediante autenticación de SQL. ...
517	2015-11-19 15:55:47.750	Logon	Error: 18456, gravedad: 14, estado: 58.
518	2015-11-19 15:55:47.750	Logon	Login failed for user 'sa'. Motivo: error al intentar iniciar sesión mediante autenticación de SQL. ...
519	2015-11-19 15:55:47.750	Logon	Error: 18456, gravedad: 14, estado: 58.
520	2015-11-19 15:55:47.750	Logon	Login failed for user 'sa'. Motivo: error al intentar iniciar sesión mediante autenticación de SQL. ...
521	2015-11-19 15:55:47.750	Logon	Error: 18456, gravedad: 14, estado: 58.
522	2015-11-19 15:55:47.750	Logon	Login failed for user 'sa'. Motivo: error al intentar iniciar sesión mediante autenticación de SQL. ...
523	2015-11-19 15:55:47.750	Logon	Error: 18456, gravedad: 14, estado: 58.

Figura 96. xp_readerrorlog ataque 3 MS SQL Server

El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]
El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]
El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]
El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]
El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]
El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]
El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]
El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]
El servidor está configurado sólo para la autenticación de Windows. [CLIENTE: 192.168.2.11]

Figura 97. xp_readerrorlog ataque 3 MS SQL Server

Y como se puede apreciar por este método también, se han intentado una serie de inicios de sesión en SQL Server el 19/11/2015 desde las 15:55, con el usuario “sa” y se ha intentado este acceso desde la maquina con IP 192.168.2.11

3.9.4.2. Análisis sobre SGBD con auditoría

Se han establecido tres métodos para auditar eventos: auditoría, triggers y SQL Profiler. Los resultados sobre el ataque sufrido con los distintos métodos son los siguientes:

a) Mediante Auditoría

Analizando la auditoría podemos comprobar que se produjeron intentos de acceso no autorizados el día 19/11/2015 a las 15:55 (el registro de auditoría marca una hora menos).

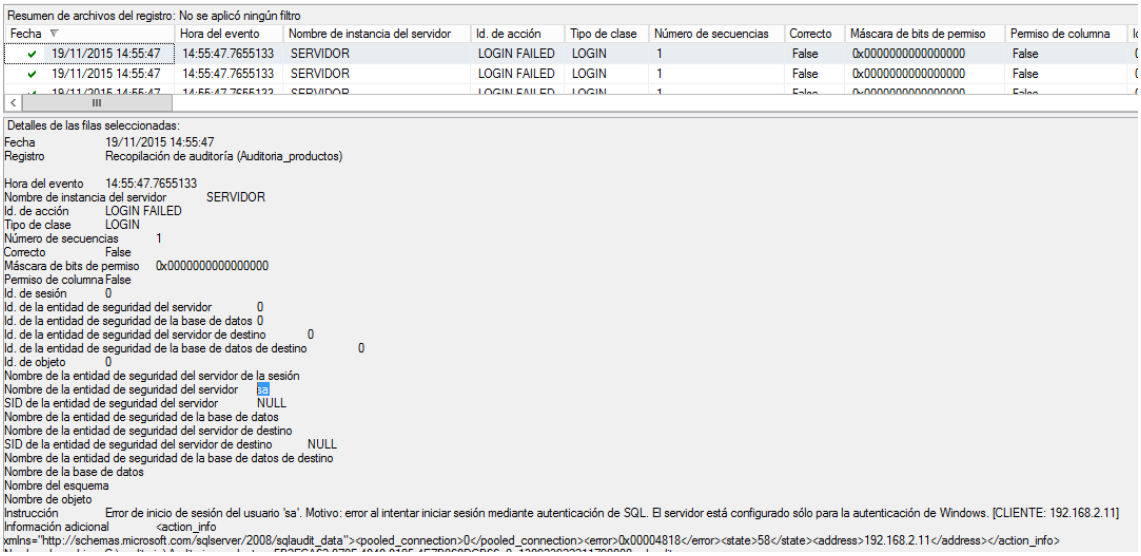


Figura 98. Auditoría ataque 4 MS SQL Server

En este caso, se ha intentado acceder a la base de datos con un ataque de fuerza bruta mediante el usuario “sa” y desde la maquina con IP 192.168.2.11.

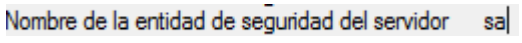


Figura 99. Auditoría ataque 4 MS SQL Server

Figura 100. Auditoría ataque 4 MS SQL Server

b) Mediante Triggers

Para este segundo modo en el que existe una auditoría mediante disparadores o triggers, el siguiente paso será analizar la tabla auxiliar donde se auditan todos los eventos realizados sobre la tabla a investigar.

Pero en este caso, los desencadenadores únicamente registran las transacciones en las que existe algún tipo de modificación sobre los datos. Y en el caso de este ataque, no se ha

producido ninguna alteración de datos, simplemente se ha intentado acceder a la base de datos con un ataque de fuerza bruta para descubrir la contraseña de “sa”.

Si esta contraseña es descubierta y se accede a la base de datos para realizar cualquier tipo de modificación, entonces si se registran estos datos mediante los desencadenadores.

c) Mediante SQL Server Profiler

Visualizando los rastros de los eventos en SQL Server Profiler podemos comprobar que se produjeron intentos de inicio de sesión el día 19/11/2015 a las 15:55 y que estos fueron rechazados.

EventClass	TextData	NTUserName	LoginName	C...	R...	Wr...	D...	Clien...	SPID	StartTime
RPC:Completed	exec sp_reset_conn...	ReportServer	NT SERVICE\ReportServer	0	0	0	0	1440	71	2015-11-19 15:55:47.153
Audit Login	-- network protoco...	ReportServer	NT SERVICE\ReportServer					1440	71	2015-11-19 15:55:47.157
RPC:Completed	declare @p1 nvarch...	ReportServer	NT SERVICE\ReportServer	0	4	0	0	1440	71	2015-11-19 15:55:47.153
Audit Logout		ReportServer	NT SERVICE\ReportServer	0	...	0	...	1440	71	2015-11-19 15:55:47.157
RPC:Completed	exec sp_reset_conn...	ReportServer	NT SERVICE\ReportServer	0	0	0	0	1440	71	2015-11-19 15:55:48.623
Audit Login	-- network protoco...	ReportServer	NT SERVICE\ReportServer					1440	71	2015-11-19 15:55:48.623
RPC:Completed	exec GetMyRunningJ...	ReportServer	NT SERVICE\ReportServer	0	2	0	0	1440	71	2015-11-19 15:55:48.623

Figura 101. Server Profiler ataque 4 MS SQL Server

Mediante este método no vemos desde donde intentaron acceder al servidor SQL Server

3.9.5. Resultados

El objetivo del análisis sobre los ataques sufridos era intentar responder a cuatro preguntas:

9. ¿Que se ha alterado?
10. ¿Cómo se ha alterado?
11. ¿Quién o desde donde lo ha alterado?
12. ¿Cuándo lo ha alterado?

En la siguiente tabla se muestra si se han alcanzado los objetivos de descubrir los ataques sufridos con los diferentes análisis.

Tipo de Análisis		Ataque 1	Ataque 2	Ataque 3	Ataque 4
Log • Ataques 1 a 3:ApexSQL Log • Ataque 4: Visor de eventos	¿Qué?	✓	✓	✗	✓
	¿Cómo?	✓	✓	✗	✓
	¿Quién?	✓	✓	✗	✓
	¿Cuándo?	✓	✓	✗	✓
Auditoría	¿Qué?	✓	✓	✓	✓
	¿Cómo?	✓	✓	✓	✓
	¿Quién?	✓	✓	✗	✓
	¿Cuándo?	✓	✓	✓	✓
Trigger	¿Qué?	✓	✓	✗	✗
	¿Cómo?	✓	✓	✗	✗
	¿Quién?	✓	✓	✗	✗
	¿Cuándo?	✓	✓	✗	✗
SQL Server Profiler	¿Qué?	✓	✓	✓	✓
	¿Cómo?	✓	✓	✓	✓
	¿Quién?	✓	✓	✗	✗
	¿Cuándo?	✓	✓	✓	✓

Tabla 8. Resultados MS SQL Server

3.10. Análisis Oracle

Como se ha comentado anteriormente, el análisis se realizara sobre dos modos diferentes del gestor de base de datos, por un lado cuando únicamente se registran logs y por otro cuando además existe establecida una auditoría sobre la base de datos.

La auditoría a su vez, podrá ser la estándar o la basada en políticas FGA

3.10.1. Análisis ataque 1

3.10.1.1. Análisis sobre SGBD sin auditoría

En este caso el análisis se tiene que realizar sobre los ficheros de redo log que registran los cambios realizados en la base de datos como resultado de transacciones o acciones internas del servidor Oracle.

Una vez localizados los registros Redo Logs en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se modifica el precio al producto.

Utilizaremos la utilidad LogMiner para analizar estos registros. Estos pasos los podemos realizar desde PLSQL o desde SQL Developer. Lo primero será indicarle a logminer los ficheros redo que necesitamos cargar, primero los localizamos:

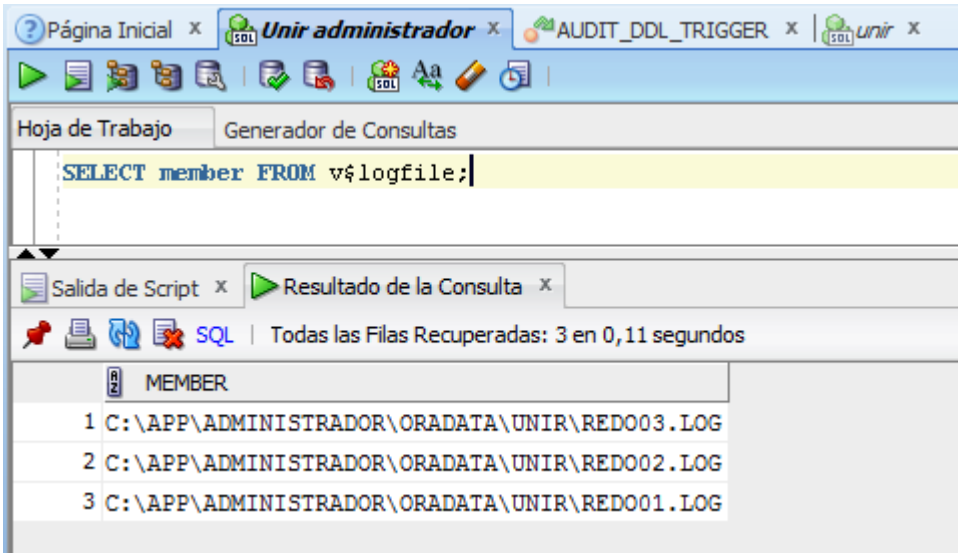


Figura 102. Redo Log LogMiner ataque 1 Oracle

Y posteriormente los cargamos a LogMiner

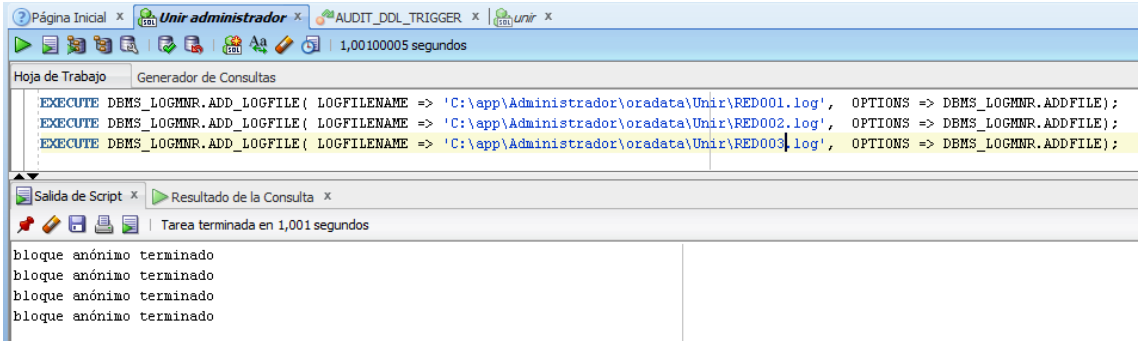


Figura 103. Carga Redo Log LogMiner ataque 1 Oracle

Ahora creamos el diccionario para que los datos mostrados los podamos interpretar.

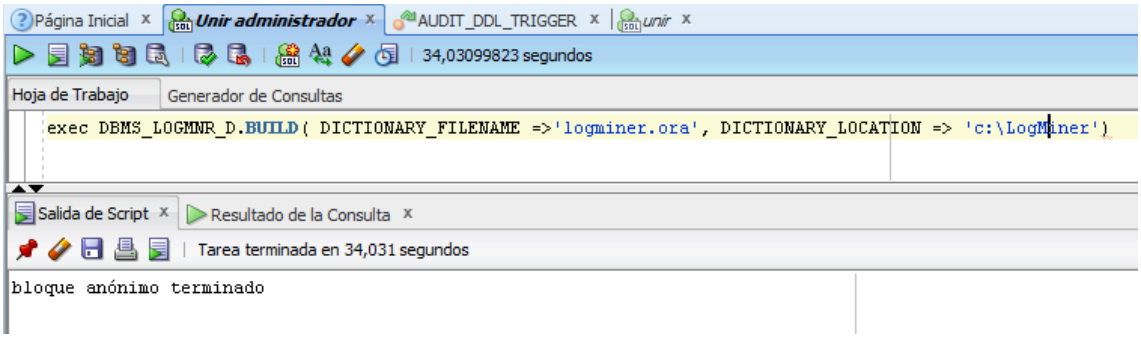


Figura 104. Diccionario LogMiner ataque 1 Oracle

Y a continuación arrancamos LogMiner con el diccionario creado en el paso anterior:

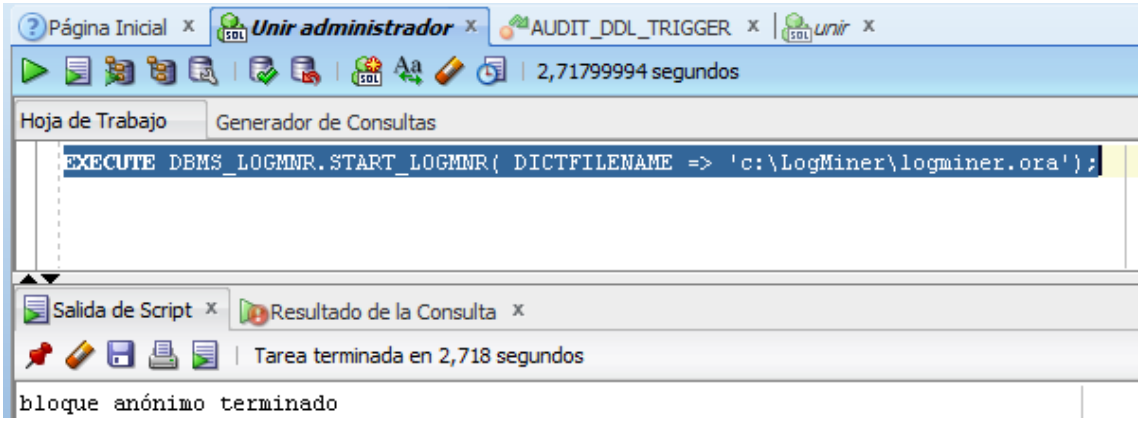


Figura 105.Inicio LogMiner ataque 1 Oracle

Ahora llega el momento de analizar los datos y buscar quien y cuando cambio el precio del producto. Para ello buscamos en la tabla creada por LogMiner LOGMNR_CONTENTS

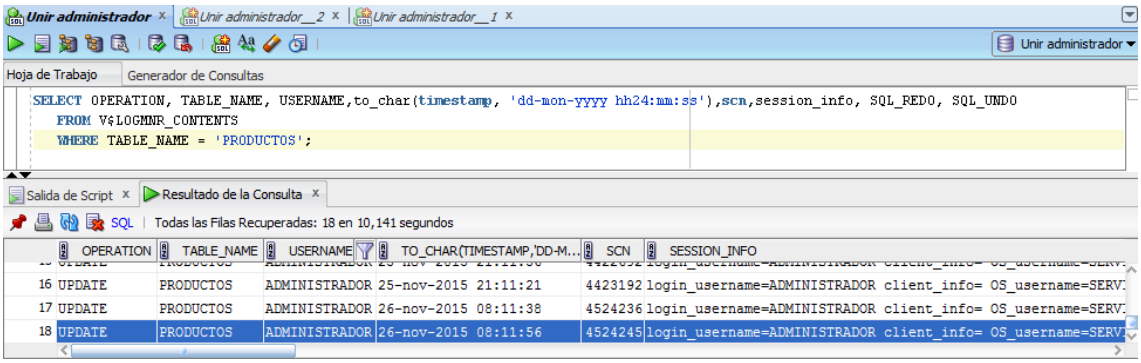


Figura 106. LogMiner ataque 1 Oracle

Podemos observar que el cambio del precio se produjo el día 26/11/2015 a las 08:11:56

Figura 107. LogMiner ataque 1 Oracle

El cambio lo realizo el usuario “administrador” desde el mismo SERVIDOR

Figura 108. LogMiner ataque 1 Oracle

Y la sentencia que ejecuto desde el programa SQLPlus

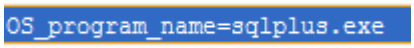


Figura 109. LogMiner ataque 1 Oracle

Fue:

Figura 110. Sentencia LogMiner ataque 1 Oracle

3.10.1.2. Análisis sobre SGBD con auditoría

Tenemos dos tipos de auditoría principal en Oracle, la auditoría estándar (Standard Auditing) y la auditoría de grano fino FGA (Fine Grained Auditing).

a) Análisis sobre SGBD con auditoría estándar

Standard auditing nos permite tener un registro de información acerca de una operación hecha en base de datos (información como usuario de ejecución, fecha, terminal).

Las operaciones de la auditoría estándar que nos interesan en este caso se registran en la tabla DBA_AUDIT_TRAIL

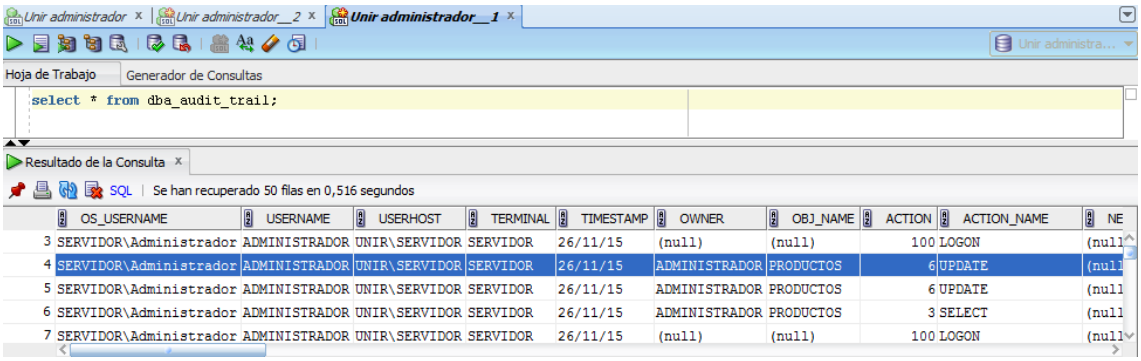


Figura 111. Auditoría estándar ataque 1 Oracle

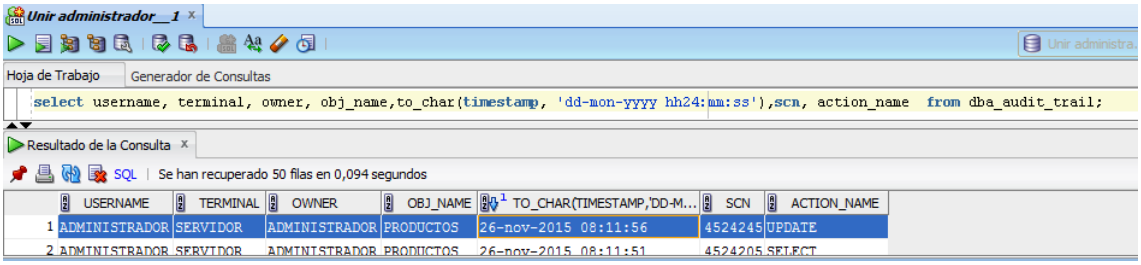


Figura 112. Auditoría estándar ataque 1 Oracle

En la imagen anterior se puede observar que el día 26/11/2015 a las 08:11:56, se realizo una operación de UPDATE sobre la tabla PRODUCTOS por parte del usuario ADMINISTRADOR desde el terminal SERVIDOR

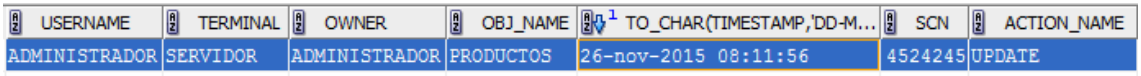


Figura 113. Auditoría estándar ataque 1 Oracle

Se puede comprobar que el numero de SCN (System Change Number) es el 4524245 que coincide con el SCN del analisis sin auditoría realizado sobre la tabla LOGMNR_CONTENTS de logMiner

El analisis sobre la auditoría estándar no proporciona mas datos. En este caso no sabemos que tipo de actualizacion se realizo. Sabemos cuando, sobre que tabla y que tipo de acccion, pero no sabemos la sentencia SQL que se ejecuto. Para saber este ultimo datos tendriamos que utilizar la auditoría de grano fino FGA

b) Análisis sobre SGBD con auditoría de grano fino FGA

FGA nos permite registrar operaciones en base a políticas de auditoría, como acciones sobre campos específicos, horarios de ejecución, y condiciones de una ejecución.

Las operaciones de la auditoría FGA que nos interesan en este caso se registran en la tabla DBA_FGA_AUDIT_TRAIL

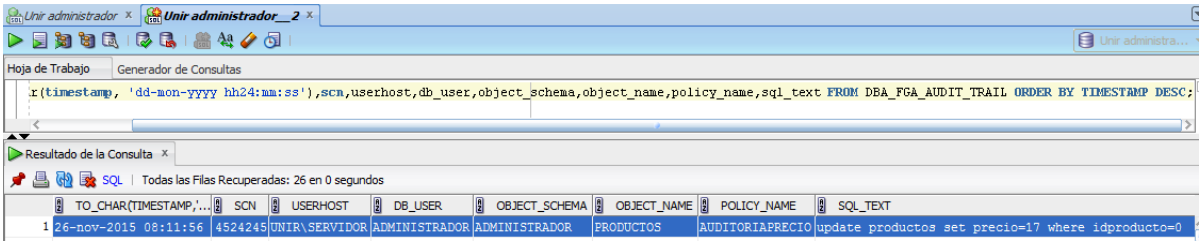


Figura 114. Auditoría FGA ataque 1 Oracle

En la imagen anterior se puede observar que el día 26/11/2015 a las 08:11:56, se realizo una operación sobre la tabla PRODUCTOS por parte del usuario ADMINISTRADOR desde el host SERVIDOR.

TO_CHAR(TIMESTAMP,'...	SCN	USERHOST	DB_USER	OBJECT_SCHEMA	OBJECT_NAME
26-nov-2015 08:11:56	4524245	UNIR\SERVIDOR	ADMINISTRADOR	ADMINISTRADOR	PRODUCTOS

Figura 115. Auditoría FGA ataque 1 Oracle

Se puede observar también la sentencia exacta que se ejecuto y la política sobre la que se registran estos sucesos de auditoría FGA:

POLICY_NAME	SQL_TEXT
AUDITORIAPRECIO	update productos set precio=17 where idproducto=0

Figura 116. Auditoría FGA ataque 1 Oracle

Se puede comprobar que el numero de SCN (System Change Number) es el 4524245 que coincide con el SCN del analisis sin auditoría realizado sobre la tabla LOGMNR_CONTENTS de logMiner y con el SCN de la auditoría estándar.

3.10.2. Análisis ataque 2

3.10.2.1. Análisis sobre SGBD sin auditoría

En este caso el análisis se tiene que realizar sobre los ficheros de redo log que registran los cambios realizados en la base de datos como resultado de transacciones o acciones internas del servidor Oracle.

Una vez localizados los registros Redo Logs en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se modifica el precio al producto.

Utilizaremos la utilidad LogMiner para analizar estos registros. Los pasos a realizar serán:

1- Cargar los Redo Logs
2- Crear el diccionario de Logminer
3- Iniciar Logminer con el diccionario creado.

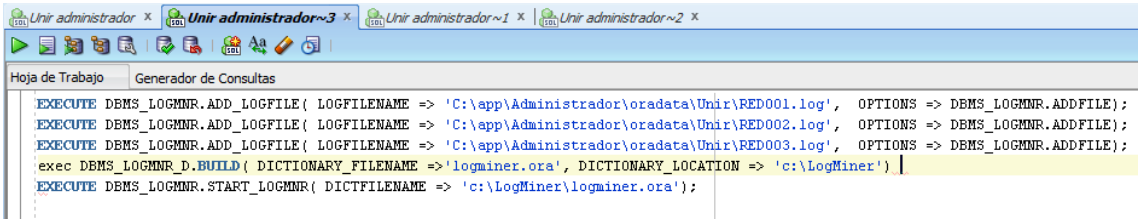


Figura 117. Carga LogMiner ataque 2 Oracle

Una vez realizados los tres pasos anteriores, llega el momento de analizar los datos y buscar quien y cuando cambio el precio del producto. Para ello buscamos en la tabla creada por LogMiner LOGMNR_CONTENTS

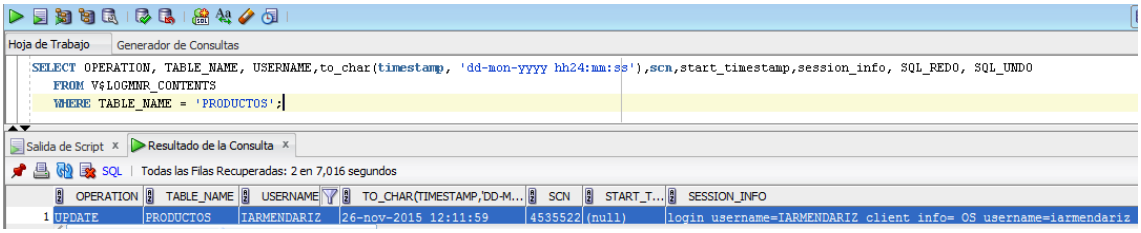


Figura 118. LogMiner ataque 2 Oracle

Podemos observar que el cambio del precio se produjo el día 26/11/2015 a las 12:11:59

OPERATION	TABLE_NAME	USERNAME	TO_CHAR(TIMESTAMP,'DD-M...	SCN
UPDATE	PRODUCTOS	IARMENDARIZ	26-nov-2015 12:11:59	4535522

Figura 119. LogMiner ataque 2 Oracle

El cambio lo realizo el usuario “iarmendariz” desde una maquina llamada W7 perteneciente al grupo de red WORKGROUP

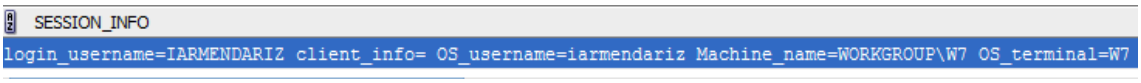


Figura 120. LogMiner ataque 2 Oracle

Se utilizo Microsoft Access

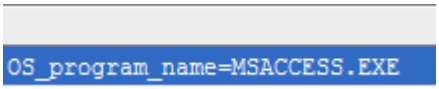


Figura 121. LogMiner ataque 2 Oracle

Y la sentencia ejecutada fue:

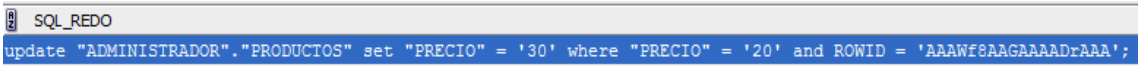


Figura 122. LogMiner ataque 2 Oracle

3.10.2.2. Análisis sobre SGBD con auditoría

Tenemos dos tipos de auditoría principal en Oracle, la auditoría estándar (Standard Auditing) y la auditoría de grano fino FGA (Fine Grained Auditing).

a) Análisis sobre SGBD con auditoría estándar

Standard auditing nos permite tener un registro de información acerca de una operación hecha en base de datos (información como usuario de ejecución, fecha, terminal).
Las operaciones de la auditoría estándar que nos interesan en este caso se registran en la tabla DBA_AUDIT_TRAIL

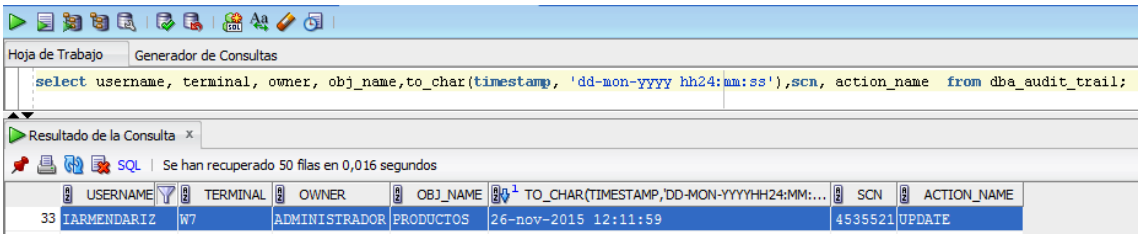
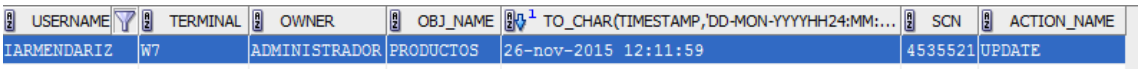


Figura 123. Auditoría estándar ataque 2 Oracle

En la imagen anterior se puede observar que el día 26/11/2015 a las 12:11:59, se realizo

una operación de UPDATE sobre la tabla PRODUCTOS por parte del usuario IARMENDARIZ desde el terminal W7



USERNAME	TERMINAL	OWNER	OBJ_NAME	TO_CHAR(TIMESTAMP,'DD-MON-YYYYHH24:MM:...	SCN	ACTION_NAME
IARMENDARIZ	W7	ADMINISTRADOR	PRODUCTOS	26-nov-2015 12:11:59	4535521	UPDATE

Figura 124. Auditoría estándar ataque 2 Oracle

El analisis sobre la auditoría estándar no proporciona mas datos. En este caso no sabemos que tipo de actualizacion se realizo. Sabemos cuando, sobre que tabla y que tipo de acccion, pero no sabemos la sentencia SQL que se ejecuto. Para saber este ultimo datos tendriamos que utilizar la auditoría de grano fino FGA

b) Análisis sobre SGBD con auditoría de grano fino FGA

FGA nos permite registrar operaciones en base a políticas de auditoría, como acciones sobre campos específicos, horarios de ejecución, y condiciones de una ejecución.

Las operaciones de la auditoría FGA que nos interesan en este caso se registran en la tabla DBA_FGA_AUDIT_TRAIL

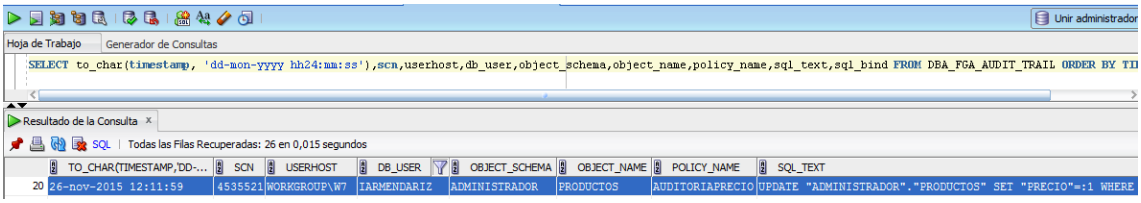
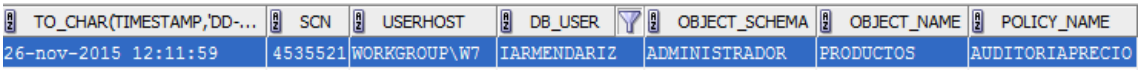


Figura 125. Auditoría FGA ataque 2 Oracle

En la imagen anterior y posterior se puede observar que el día 26/11/2015 a las 12:11:59, se realizo una operación sobre la tabla PRODUCTOS por parte del usuario IARMENDARIZ desde el host W7.



TO_CHAR(TIMESTAMP,'DD-...	SCN	USERHOST	DB_USER	OBJECT_SCHEMA	OBJECT_NAME	POLICY_NAME
26-nov-2015 12:11:59	4535521	WORKGROUP\W7	IARMENDARIZ	ADMINISTRADOR	PRODUCTOS	AUDITORIAPRECIO

Figura 126. Auditoría FGA ataque 2 Oracle

Se puede observar también la sentencia exacta que se ejecuto, teniendo en cuenta que los datos aparecen en variables:

Figura 127. Auditoría FGA ataque 2 Oracle

Y las variables se pueden ver en la siguiente imagen:

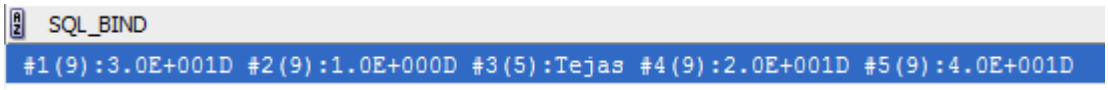


Figura 128. Auditoría FGA ataque 2 Oracle

La interpretación de las variables es la siguiente:

Variable	Numero de caracteres	Valor	Valor Real
#1	(9)	3.0E+001D	30
#2	(9)	1.0E+000D	1
#3	(5)	Tejas	Tejas
#4	(9)	2.0E+001D	20
#5	(9)	4.0E+001D	40

Tabla 9. Variables FGA Oracle

La sentencia ejecutada ha sido:

UPDATE ADMINISTRADOR.PRODUCTOS SET PRECIO=#1 WHERE IDPRODUCTO=#2 AND PRODUCTO=#3 AND PRECIO=#4 AND STOCK=#5

Traduciendo las variables por su valor:

UPDATE ADMINISTRADOR.PRODUCTOS SET PRECIO=30 WHERE IDPRODUCTO=1 AND PRODUCTO=’ Tejas’ AND PRECIO=20 AND STOCK=40

3.10.3. Análisis ataque 3

3.10.3.1. Análisis sobre SGBD sin auditoría

En este caso el análisis se tiene que realizar sobre los ficheros de redo log que registran los cambios realizados en la base de datos como resultado de transacciones o acciones internas del servidor Oracle.

Una vez localizados los registros Redo Logs en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se modifica el precio al producto.

Utilizaremos la utilidad LogMiner para analizar estos registros. Los pasos a realizar serán:

1- Cargar los Redo Logs
2- Crear el diccionario de Logminer
3- Iniciar Logminer con el diccionario creado.

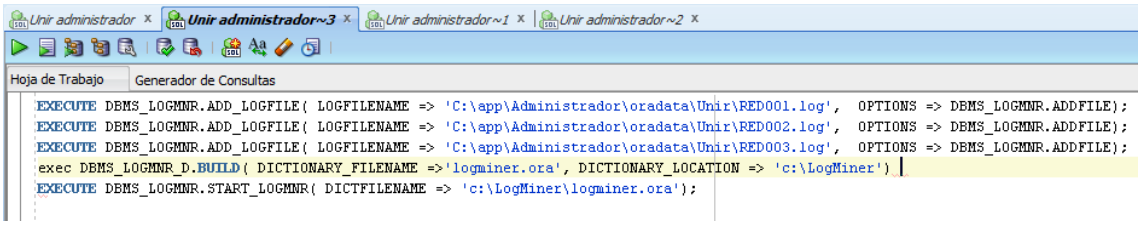


Figura 129. Carga LogMiner ataque 3 Oracle

Una vez realizados los tres pasos anteriores, llega el momento de analizar los datos y buscar quien y cuando cambio el precio del producto. Para ello buscamos en la tabla creada por LogMiner LOGMNR_CONTENTS

Pero en este caso, los Redo Logs únicamente están registrando las transacciones en las que existe algún tipo de modificación sobre los datos. Y en el caso de este ataque, no se ha producido ninguna alteración de datos, simplemente se ha tenido acceso a ellos, se ha podido producir un robo de datos.

3.10.3.2. Análisis sobre SGBD con auditoría

a) Análisis sobre SGBD con auditoría estándar

Standard auditing nos permite tener un registro de información acerca de una operación hecha en base de datos (información como usuario de ejecución, fecha, terminal).
Las operaciones de la auditoría estándar que nos interesan en este caso se registran en la tabla DBA_AUDIT_TRAIL

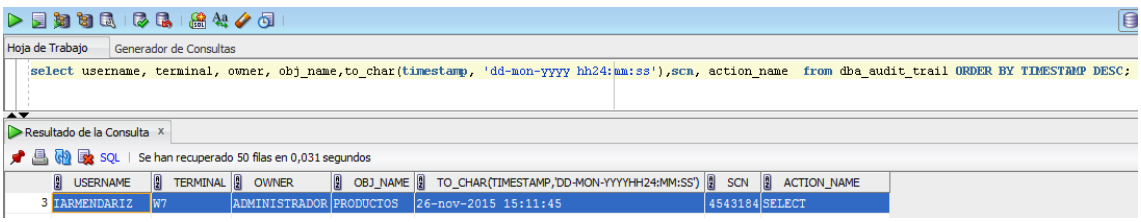


Figura 130. Auditoría estándar ataque 3 Oracle

En la imagen anterior se puede observar que el día 26/11/2015 a las 15:11:45, se realizo una operación de SELECT sobre la tabla PRODUCTOS por parte del usuario IARMENDARIZ desde el terminal W7

USERNAME	TERMINAL	OWNER	OBJ_NAME	TO_CHAR(TIMESTAMP,'DD-MON-YYYYHH24:MM:SS')	SCN	ACTION_NAME
IARMENDARIZ	W7	ADMINISTRADOR	PRODUCTOS	26-nov-2015 15:11:45	4543184	SELECT

Figura 131. Auditoría estándar ataque 3 Oracle

El analisis sobre la auditoría estándar no proporciona mas datos. En este caso no sabemos que tipo de actualizacion se realizo. Sabemos cuando, sobre que tabla y que tipo de acccion, pero no sabemos la sentencia SQL que se ejecuto. Para saber este ultimo datos tendriamos que utilizar la auditoría de grano fino FGA

b) Análisis sobre SGBD con auditoría de grano fino FGA

FGA nos permite registrar operaciones en base a políticas de auditoría, como acciones sobre campos específicos, horarios de ejecución, y condiciones de una ejecución.

Las operaciones de la auditoría FGA que nos interesan en este caso se registran en la tabla DBA_FGA_AUDIT_TRAIL

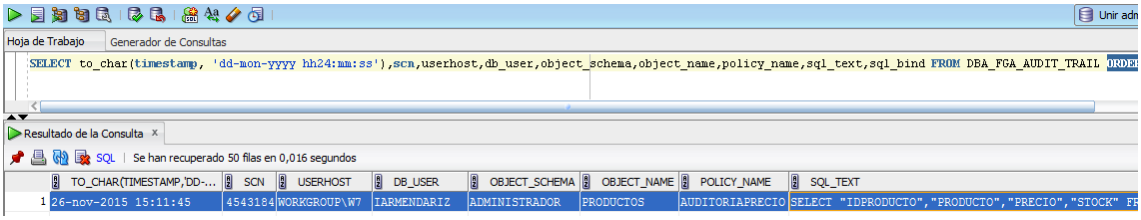


Figura 132. Auditoría FGA ataque 3 Oracle

En la imagen anterior y posterior se puede observar que el día 26/11/2015 a las 15:11:45, se realizo una operación sobre la tabla PRODUCTOS por parte del usuario IARMENDARIZ desde el host W7.

TO_CHAR(TIMESTAMP,'DD-...	SCN	USERHOST	DB_USER	OBJECT_SCHEMA	OBJECT_NAME	POLICY_NAME
26-nov-2015 15:11:45	4543184	WORKGROUP\W7	IARMENDARIZ	ADMINISTRADOR	PRODUCTOS	AUDITORIAPRECIO

Figura 133. Auditoría FGA ataque 3 Oracle

Se puede observar también la sentencia exacta que se ejecuto, teniendo en cuenta que los datos aparecen en variables:

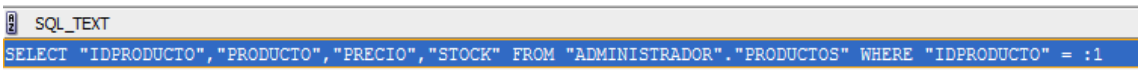


Figura 134. Auditoría FGA ataque 3 Oracle

Y las variables se pueden ver en la siguiente imagen:

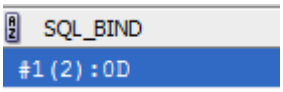


Figura 135. Auditoría FGA ataque 3 Oracle

En el caso de este ataque, en el que la sentencia introducida por el atacante es:

Select * from productos where idproducto= 999 OR 1=1

Se está traduciendo por:

Select * from productos where idproducto= 0

3.10.4. Análisis ataque 4

3.10.4.1. Análisis sobre SGBD sin auditoría

En este caso el análisis se tiene que realizar sobre los ficheros de redo log que registran los cambios realizados en la base de datos como resultado de transacciones o acciones internas del servidor Oracle.

Una vez localizados los registros Redo Logs en la imagen del servidor obtenida en la recopilación de evidencias, tendremos que analizarlo registro a registro hasta encontrar la evidencia donde se modifica el precio al producto.

Utilizaremos la utilidad LogMiner para analizar estos registros. Los pasos a realizar serán:

1- Cargar los Redo Logs
2- Crear el diccionario de Logminer
3- Iniciar Logminer con el diccionario creado.

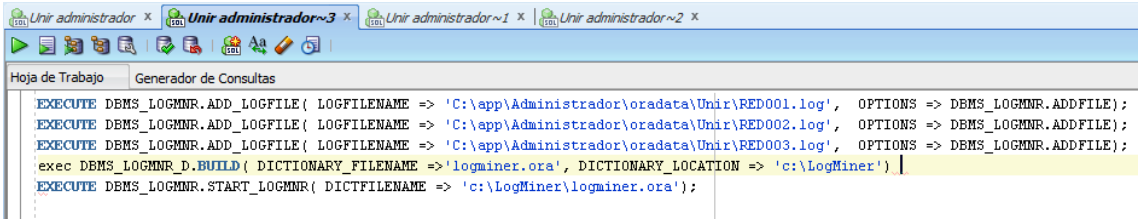


Figura 136. Carga LogMiner ataque 4 Oracle

Una vez realizados los tres pasos anteriores, llega el momento de analizar los datos y buscar quien y cuando cambio el precio del producto. Para ello buscamos en la tabla creada por LogMiner LOGMNR_CONTENTS

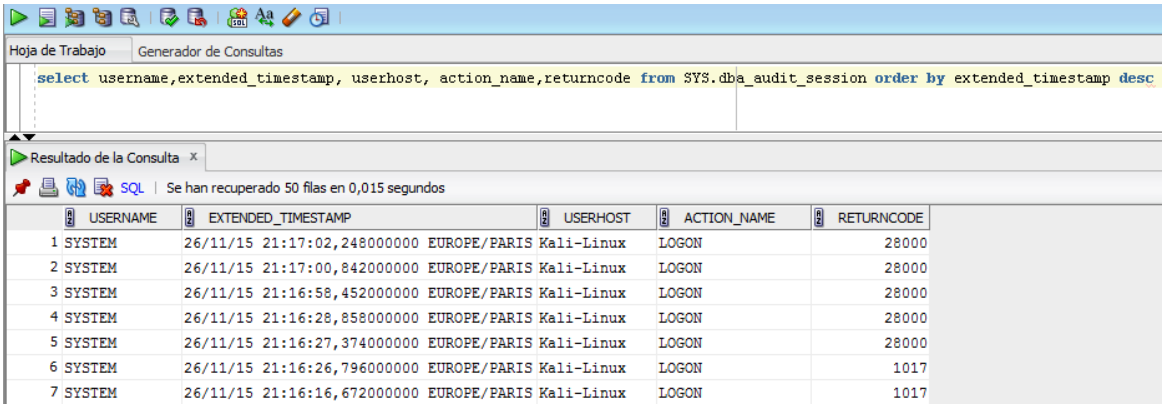
Pero en este caso, los Redo Logs únicamente están registrando las transacciones en las que existe algún tipo de modificación sobre los datos. Y en el caso de este ataque, no se ha producido ninguna alteración de datos, simplemente se ha intentado acceder a la base de datos con un ataque de fuerza bruta para descubrir la contraseña de system.

Si esta contraseña es descubierta y se accede a la base de datos para realizar cualquier tipo de modificación, entonces si se registran estos datos en la auditoría.

3.10.4.2. Análisis sobre SGBD con auditoría

1. Análisis sobre SGBD con auditoría estándar

Standard auditing nos permite tener un registro de información acerca de una operación hecha en base de datos (información como usuario de ejecución, fecha, terminal).
Las operaciones de la auditoría estándar que nos interesan en este caso se registran en la tabla DBA_AUDIT_SESSION



	USERNAME	EXTENDED_TIMESTAMP	USERHOST	ACTION_NAME	RETURNCODE
1	SYSTEM	26/11/15 21:17:02,248000000 EUROPE/PARIS	Kali-Linux	LOGON	28000
2	SYSTEM	26/11/15 21:17:00,842000000 EUROPE/PARIS	Kali-Linux	LOGON	28000
3	SYSTEM	26/11/15 21:16:58,452000000 EUROPE/PARIS	Kali-Linux	LOGON	28000
4	SYSTEM	26/11/15 21:16:28,858000000 EUROPE/PARIS	Kali-Linux	LOGON	28000
5	SYSTEM	26/11/15 21:16:27,374000000 EUROPE/PARIS	Kali-Linux	LOGON	28000
6	SYSTEM	26/11/15 21:16:26,796000000 EUROPE/PARIS	Kali-Linux	LOGON	1017
7	SYSTEM	26/11/15 21:16:16,672000000 EUROPE/PARIS	Kali-Linux	LOGON	1017

Figura 137. Auditoría estándar ataque 4 Oracle

En la imagen anterior podemos observar cómo se intenta acceder a Oracle con el usuario system y probando múltiples contraseñas. El ataque se produce el 26/11/2015 hacia las 21:16.

Después de varios intentos de inicio de sesión probando distintas contraseñas (código de retorno 1017), el usuario system se ha bloqueado a las 21:16:27 (código de retorno 28000)

USERNAME	EXTENDED_TIMESTAMP	USERHOST	ACTION_NAME	RETURNCODE
SYSTEM	26/11/15 21:16:58,452000000 EUROPE/PARIS	Kali-Linux	LOGON	28000
SYSTEM	26/11/15 21:16:28,858000000 EUROPE/PARIS	Kali-Linux	LOGON	28000
SYSTEM	26/11/15 21:16:27,374000000 EUROPE/PARIS	Kali-Linux	LOGON	28000
SYSTEM	26/11/15 21:16:26,796000000 EUROPE/PARIS	Kali-Linux	LOGON	1017
SYSTEM	26/11/15 21:16:16,672000000 EUROPE/PARIS	Kali-Linux	LOGON	1017

Figura 138. Auditoría estándar ataque 4 Oracle

En la tabla siguiente podemos observar los códigos de retorno para los distintos tipos de inicio de sesión posibles

Código de retorno	Valor
00911	'Invalid Character'
00988	'Missing or invalid password(s)'
01004	'Logon denied.'
01005	'Null Password'
01017	'Invalid username/password.'
01031	'No Privilege'
01045	'User string lacks CREATE SESSION privilege; logon denied.'
01918	'No Such UserID'
01920	'No Such Role'
09911	'Incorrect user password.'
28000	'The account is locked.'
28001	'Your password has expired.'
28002	'Your account will expire soon; change your password now.'
28003	'The password is not complex enough.'
28007	'Password cannot be reused.'
28008	'Invalid old password.'
28009	'Connection to sys should be as sysdba or sysoper.'
28011	'Your account will expire soon; change your password now.'
28221	'The original password was not supplied.'

Tabla 10. Códigos de retorno Oracle

2. Análisis sobre SGBD con auditoría de grano fino FGA

FGA nos permite registrar operaciones en base a políticas de auditoría, como acciones sobre campos específicos, horarios de ejecución, y condiciones de una ejecución.

En este caso, la política aplicada para registrar la auditoría FGA no registra los intentos de sesión fallidos, por lo que por este punto no podremos obtener ningún tipo de información.

3.10.5. Resultados

El objetivo del análisis sobre los ataques sufridos era intentar responder a cuatro preguntas:

13. ¿Que se ha alterado?
14. ¿Cómo se ha alterado?
15. ¿Quién o desde donde lo ha alterado?
16. ¿Cuándo lo ha alterado?

En la siguiente tabla se muestra si se han alcanzado los objetivos de descubrir los ataques sufridos con los diferentes análisis.

Tipo de Análisis		Ataque 1	Ataque 2	Ataque 3	Ataque 4
Log LogMiner	¿Qué?	✓	✓	✗	✗
	¿Cómo?	✓	✓	✗	✗
	¿Quién?	✓	✓	✗	✗
	¿Cuándo?	✓	✓	✗	✗
Auditoría Estándar	¿Qué?	✗	✗	✗	✓
	¿Cómo?	✓	✓	✓	✓
	¿Quién?	✓	✓	✓	✓
	¿Cuándo?	✓	✓	✓	✓
Auditoría FGA	¿Qué?	✓	✓	✓	✗
	¿Cómo?	✓	✓	✓	✗
	¿Quién?	✓	✓	✓	✗
	¿Cuándo?	✓	✓	✓	✗

Tabla 11. Resultados Oracle

3.11. Análisis de los resultados

Una vez realizados los análisis de los ataques sufridos sobre los cuatro gestores de bases de datos vamos a compararlos para ver las diferencias en la detección de los ataques, en la posibilidad de utilización de herramientas propias y no de terceros y en la facilidad en la obtención de los datos.

3.11.1. Detección de los ataques

En la siguiente tabla se muestra a rasgos generales los ataques detectados en cada sistema gestor de bases de datos. Para ver más a fondo que parte de cada ataque ha sido detectado en cada SGBD se puede acceder al último punto del análisis de cada gestor.

SGBD		Ataque 1	Ataque 2	Ataque 3	Ataque 4
MySQL	Log	✓	✓	✓	✓
	Trigger	✓	✓	✗	✗
PostgreSQL	Log	✓	✓	✓	✓
	Trigger	✓	✓	✗	✗
MS SQL Server	Log	✓	✓	✗	✓
	Auditoría	✓	✓	✓	✓
	Trigger	✓	✓	✗	✗
	Server Profiler	✓	✓	✓	✓
Oracle	Log	✓	✓	✗	✗
	Aud. Estándar	✓	✓	✓	✓
	Auditoría FGA	✓	✓	✓	✗

Tabla 12. Comparación detección ataques

En la tabla anterior podemos observar que combinando los distintos modos de los diferentes SGBD se detectan los cuatro ataques en los cuatro gestores.

Tanto en MySQL como en PostgreSQL, los mejores resultados los obtenemos consultando los logs, puesto que la auditoría mediante trigger únicamente detecta la alteración de los datos, y en los dos últimos ataque no se ha producido alteración de los mismos.

Habría que añadir que el análisis de los logs de MySQL y PostgreSQL únicamente muestra las sentencias SQL ejecutadas, por lo que sí es una modificación de datos no muestran el valor anterior, valor que si muestra el análisis mediante la auditoría con triggers.

Los transaction log de Microsoft SQL Server detectan tres de los cuatro ataques. Estos log únicamente detectan modificaciones y no consultas de los datos. La detección del cuarto ataque (inicios de sesión fallidos) se realiza mediante el visor de eventos de Windows.

La auditoría y SQL Server Profiler de Microsoft SQL Server detectan los cuatro ataques. Pero mediante los trigger únicamente se detectan la alteración de los datos, por lo que no detectan los dos últimos ataques.

Los redo log de Oracle únicamente registran las alteraciones de los datos (primer y segundo ataque). La auditoría estándar detecta los cuatro ataques, pero no se pueden ver las sentencias SQL ejecutadas, por lo que no sabemos los datos modificados. La de grano fino (FGA) si que muestra la sentencia ejecutada, pero en el escenario no estaba aplicada la política para detectar inicios de sesión, por lo que el último ataque no ha sido detectado.

3.11.2. Herramientas utilizadas.

En la tabla siguiente se muestran las herramientas utilizadas en el análisis de los SGBD, indicando si se trata de herramientas propias del SGBD o herramientas de terceros.

SGBD	Herramienta	Propia	Terceros	
			Libre	De Pago
MySQL	Consola	✓		
	phpMyAdmin		✓	
	Notepad		✓	
PostgreSQL	PgAdmin 3		✓	
	Notepad		✓	
MS SQL Server	Management Studio	✓		
	SQL Server Profiler	✓		
	Visor de Eventos Windows		✓	
	ApexSQL Log			✓
Oracle	SQL Plus	✓		
	SQL Developer	✓		

Tabla 13. Herramientas utilizadas

Para este trabajo se ha intentado no depender de herramientas de terceros que pueden suponer un costo en la licencia y utilizar las propias herramientas de cada SGBD o herramientas propias del sistema operativo (notepad y visor de eventos de Windows).

Salvo para la lectura de los transaction logs de Microsoft SQL Server, en el que se ha tenido que utilizar una herramienta de pago (ApexSQL log), en los demás casos se han podido detectar y analizar todos los ataques mediante herramientas propias o de software libre (phpMyAdmin y PgAdmin 3).

En el caso de los transaction logs de MS SQL Server, interpretarlos es una tarea muy compleja porque son columnas ilegibles que habrá que traducir y no hay documentación oficial disponible para estos logs. Es por ello que se ha optado por utilizar una herramienta de terceros y de pago, en este caso no hay herramientas de que analicen estos logs de software libre.

He utilizado también las herramientas libres phpMyAdmin y PgAdmin 3 para analizar MySQL y PostgreSQL respectivamente por ser herramientas graficas y facilitar la tarea, pero se podrían obtener los mismos resultados mediante sus respectivas consolas.

3.11.3. Sencillez en la detección

En la tabla inferior voy a puntuar la sencillez en la detección de los ataques, teniendo en cuenta las facilidades que proporciona cada gestor de bases de datos en analizar los logs, en crear las auditorías y analizarlas, etc.

SGBD		¿Requiere programación?	¿Requiere configuración?	Dificultad
MySQL	Log	No	No	Baja
	Trigger	Si	Si	Media
PostgreSQL	Log	No	No	Baja
	Trigger	Si	Si	Media
MS SQL Server	Log	No	No	Alta
	Auditoría	No	Si	Media-Baja
	Trigger	Si	Si	Media
	Server Profiler	No	Si	Media
Oracle	Log	No	No	Media
	Aud. Estándar	No	Si	Media-Baja
	Auditoría FGA	No	Si	Media

Tabla 14. Sencillez en la detección

Como se puede observar en la tabla anterior, la única nota alta en nivel de dificultad la recibe el análisis de los transaction logs de Microsoft SQL Server. Esta nota se refiere al análisis en bruto de los logs, sin utilizar herramientas de terceros como ApexSQL Log. La nota utilizando esta herramienta seria de media-baja.

4. Conclusiones

En este trabajo se ha pretendido exponer los fundamentos básicos de un análisis forense, así como sus objetivos.

Se ha tratado la seguridad en sistemas gestores de bases de datos, tanto comerciales como de código libre. Profundizando en los gestores MySQL, PostgreSQL, Microsoft SQL Server y Oracle, así como en su seguridad e implantación de auditorías.

En el aspecto experimental, se han realizado cuatro ataques distintos sobre los cuatro gestores de bases de datos arriba mencionados. Dos de esos ataques realizados por personal interno de la organización y los otros dos por personas ajenas a la organización realizando SQL Injection y ataque de fuerza bruta.

Una vez realizados los ataques y adquiridas las evidencias, se han analizado intentando no utilizar para ello herramientas de terceros sino valiéndonos de las propias utilidades de los gestores cuando ha sido posible.

Con el análisis de cada uno de los ataques a los distintos SGBD hemos intentado responder a las cuatro preguntas fundamentales del análisis forense:

- ¿Qué se ha alterado?
- ¿Cómo se ha alterado?
- ¿Quién lo ha alterado?
- ¿Cuándo se ha alterado?

En los cuatro SGBD hemos conseguido detectar todos los ataques realizados, obteniendo distinta información dependiendo del análisis realizado (logs, auditoría, triggers, etc.).

Salvo en el caso de los logs de Microsoft SQL Server en el que se ha utilizado una herramienta de terceros y de pago, en los demás casos el análisis se ha llevado a cabo mediante las utilidades de las propios SGBD o herramientas libres o disponibles en el sistema operativo.

Por último se han comparado los resultados valiéndonos de tres métricas diferentes:

1. Detección de los ataques.
2. Herramientas utilizadas.
3. Sencillez en la detección de los ataques.

Como futuras líneas de trabajo podríamos destacar la realización de más y distintos ataques sobre los SGBD. Ampliar los SGBD analizados a más productos. Utilizar herramientas de terceros y compararlas con las utilidades de los sistemas gestores

5. Bibliografía y Webgrafía

1. Instituto Nacional de Ciberseguridad (Incibe). Obtenido de: https://www.incibe.es/

2. Paul Henry. (2009). Best practices in digital evidence collection. Obtenido de: https://digital-forensics.sans.org/blog/2009/09/12/best-practices-in-digital-evidence-collection/

3. NIST National Institute of Standards and Technology. Obtenido de: http://www.nist.gov/

4. Carrión Patiño, Liliana Isabel. (2008). *Trabajo investigación manejo de la evidencia digital en el derecho informático y aplicaciones de software*. Obtenido de: http://dspace.ucacue.edu.ec/bitstream/reducacue/4012/4/Trabajo%20Investigaci%C3%B3n%20MANEJO%20DE%20LA%20EVIDENCIA%20DIGITAL%20EN%20EL%20DERECHO%20INFORMATICO%20Y%20APICACIONES%20DE%20SOFTWARE.pdf

5. José Luis Rivas López. (2009). Introducción al análisis forense. Obtenido de: http://jlrivas.webs.uvigo.es/downloads/publicaciones/Analisis%20forense%20de%20sistemas%20informaticos.pdf

6. Asier Martínez Retenaga. (2014). Guía de toma de evidencias. Obtenido de: https://www.incibe.es/extfrontinteco/img/File/intecocert/ManualesGuias/incibe_toma_evidencias_analisis_forense.pdf

7. Brezinski y T. Killalea (2002). Guidelines for Evidence Collection and Archiving http://www.ietf.org/rfc/rfc3227.txt

8. Boletín Oficial del Estado (1978). *Constitución Española.* Obtenido de: https://www.boe.es/buscar/act.php?id=BOE-A-1978-31229&b=17&tn=1&p=19781229#a13

9. Boletín Oficial del Estado (2000). *Ley de Enjuiciamiento Civil.* Obtenido de: https://www.boe.es/buscar/act.php?id=BOE-A-2000-323

10. Boletín Oficial del Estado (1999). *Ley de Protección de Datos de Carácter Personal.* Obtenido de: https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750

11. Boletín Oficial del Estado (2002). *Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico.* Obtenido de: https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758

12. Boletín Oficial del Estado (2007). *Ley de Conservación de Datos Relativos a las Comunicaciones y las Redes Públicas.* Obtenido de: https://www.boe.es/buscar/act.php?id=BOE-A-2007-18243

13. Boletín Oficial del Estado (1995). *Código Penal.* Obtenido de: https://www.boe.es/buscar/act.php?id=BOE-A-1995-25444

14. Henry, P. (2009). Best practices in digital evidence collection. Obtenido de: https://digitalforensics.sans.org/blog/2009/09/12/best-practices-in-digital-evidence-collection/

15. Una comparación de alto nivel entre Oracle y SQL Server (2015). Obtenido de: http://blog.jmacoe.com/gestion_ti/base_de_datos/una-comparacion-de-alto-nivel-entre-oracle-y-sql-server/

16. MySQL Security (2015). Obtenido de: http://dev.mysql.com/doc/refman/5.7/en/security.html

17. Visión general de la arquitectura de MySQL 5.1 (2008). Obtenido de: http://cnx.org/contents/fa33ef02-f882-4b8d-9f67-4be268dd6d03@1/Visin-general-de-la-arquitectu

18. Bases de datos y sus vulnerabilidades más comunes (2015). Obtenido de: http://www.acens.com/wp-content/images/2015/03/vulnerabilidades-bbdd-wp-acens.pdf

19. Top Ten Database Security Threats (2015). Obtenido de: http://www.imperva.com/docs/wp_topten_database_threats.pdf

20. Centro de seguridad para el motor de base de datos SQL Server (2014). Obtenido de: https://msdn.microsoft.com/es-ES/library/bb510589(v=sql.120).aspx

21. MySQL 5.6 Reference Manual-Security (2015). Obtenido de: http://dev.mysql.com/doc/refman/5.6/en/security.html

22. PostgreSQL 9.4.5 Documentation (2015). Obtenido de: http://www.postgresql.org/docs/9.4/static/index.html

23. Información general sobre seguridad de SQL Server (2015). Obtenido de: https://msdn.microsoft.com/es-es/library/bb669078(v=vs.110).aspx

24. Autenticación en SQL Server (2015). Obtenido de: https://msdn.microsoft.com/es-es/library/bb669066(v=vs.110).aspx

25. Roles de servidor y base de datos en SQL Server (2015). Obtenido de: https://msdn.microsoft.com/es-es/library/bb669065(v=vs.110).aspx

26. Propiedad y separación usuario-esquema en SQL Server (2015). Obtenido de: https://msdn.microsoft.com/es-es/library/bb669061(v=vs.110).aspx

27. Autorización y permisos en SQL Server (2015). Obtenido de: https://msdn.microsoft.com/es-es/library/bb669084(v=vs.110).aspx

28. Cifrado de datos en SQL Server (2015). Obtenido de: https://msdn.microsoft.com/es-es/library/bb669072(v=vs.110).aspx

29. Trigger genérico para auditar cambios en una tabla (2011). http://dblearner.com/trigger-genrico-para-auditar-cambios-en-una-tabla/

30. Using LogMiner to Analyze Redo Log Files (2015). Obtenido de: https://docs.oracle.com/cd/E11882_01/server.112/e22490/logminer.htm#SUTIL1558

31. Juan Javier Rodríguez Guisado (2015). Auditoría en Oracle, Sistemas Gestores de Base de Datos. Obtenido de: http://docplayer.es/4718284-Auditoria-en-oracle-sistemas-gestores-de-base-de-datos-juan-javier-rodriguez-guisado.html

32. Yannick Jaquier (2013). Fine-Grained Auditing (FGA) hands-on. Obtenido de: http://blog.yannickjaquier.com/oracle/fine-grained-auditing-fga-hands-on.html

33. DB-Engines Ranking of Relational DBMS. Obtenido de: http://db-engines.com/en/ranking/relational+dbms

34. System Properties Comparison Microsoft SQL Server vs. MySQL vs. Oracle vs. PostgreSQL. Obtenido de:http://db-engines.com/en/system/Microsoft+SQL+Server%3BMySQL%3BOracle%3BPostgreSQL

35. Mysql vs Postgres vs Oracle vs Mssql performance. Obtenido de: http://phpdao.com/mysql_postgres_oracle_mssql/

36. The High-Performance SQL Blog. Count Distinct Compared on Top 4 SQL Databases. Obtenido de: https://www.periscopedata.com/blog/count-distinct-in-mysql-postgres-sql-server-and-oracle.html