Introducción

PID_00191661

Vicente Díaz Sáez

Universitat Oberta de Catalunya

Índice

1. Seguridad en bases de datos y aplicaciones web

Las bases de datos están en todas partes. Es así de simple, son el almacén de la información de uso diario para prácticamente todo. Almacenan datos bancarios, médicos, el censo de la población, antecedentes penales, información sobre avistamientos OVNI y los datos de la declaración de la renta. No hay ninguna organización o empresa que no haga uso de un modo u otro de una base de datos, por lo que está claro que son una pieza clave y jugosa para cualquier atacante, ante la que cualquier medida de protección es poca.

Esto no sería un gran problema si nuestras bases de datos se encontraran guardadas en una caja de seguridad en un lugar desconocido, pero entonces tampoco serían muy útiles. Las bases de datos forman parte de un ecosistema desde el cual se permite el acceso a la información que contienen a ciertos usuarios. Dentro de este ecosistema casi siempre encontramos un aplicativo web, o dándole la vuelta, la práctica totalidad de servicios web utilizan de un modo u otro una base de datos. Es por ello que ambos mundos están íntimamente interrelacionados y por lo que cuando pensamos en seguridad no podemos considerar en dichos elementos como aislados.

Los servicios web y las aplicaciones que lo implementan viven en la actualidad su momento de máxima expansión. El crecimiento de los servicios 2.0 y la proliferación del *cloud computing* han supuesto la explosión de la demanda para todo tipo de aplicaciones que implementan desde la posición GPS de un usuario de un terminal inteligente hasta la generación de números aleatorios. Sea como sea, y más allá de los datos que hayan detrás, la aplicación en sí misma supone la puerta de entrada para un potencial atacante a la infraestructura en la que se encuentra. Un servicio mal securizado puede resultar en un ataque que consiga una denegación de servicio, que se cambie la página web de la compañía o que se obtenga un control total de la red interna, es decir, una amenaza para todos los clientes de la misma.

Se pueden encontrar numerosos ejemplos de ello. No es que no exista abundante documentación anterior al 2011, pero quizá dicho año ha sido especialmente significativo en lo que se refiere a la popularización de ataques dirigidos contra grandes compañías y entidades más allá del afán de lucro o de obtención de "reputación" habitual entre los *hackers* "buenos". Grupos heterogéneos como Anonymous o Lulzsec consiguieron evitar los controles de seguridad existentes en compañías y entidades de talla mundial, publicando posteriormente los datos obtenidos. Y algo muy interesante: ¡en la mayoría de los casos los ataques fueron extremadamente simples!

La pérdida o acceso no autorizado, y especialmente la publicación de los datos privados de una entidad, pueden suponer un daño irreparable para una organización en su reputación, recordemos el caso de HBGARY, empresa dedicada a la seguridad en Estados Unidos. Ante una pérdida se pueden adoptar medidas y precauciones, como copias de seguridad, pero una intrusión es un tema más delicado: el acceso a la información clave de una compañía puede suponer una pérdida irreparable.

> Imaginemos el caso de una empresa farmacéutica que dedica millones a investigación y desarrollo de nuevos fármacos. Dichos desarrollos duran años y en muchos casos suponen la supervivencia de la compañía, seguro que todos somos capaces de encontrar un ejemplo de farmacéutica que se basa en un único medicamento. Un caso de espionaje industrial en este entorno sería fatal.

No se trata únicamente de temas de espionaje, existen temas legales que hay que tener muy presentes. La LOPD (Ley Orgánica de Protección de Datos de Carácter Personal en España), en vigor desde el 13 de diciembre de 1999, hace referencia precisamente a los datos que almacenan las empresas españolas acerca de los ciudadanos y que son de carácter personal. Dicha ley establece una serie de medidas de obligado cumplimiento por parte de las organizaciones que almacenan dichos datos y en función de la información que se almacene. Se establecen tres niveles de importancia de datos y medidas relacionadas con cada uno de ellos de obligado cumplimiento, con sus correspondientes sanciones.

En el Estado de California entró en vigor en julio del 2003 la ley Orgánica SB 1386, que obliga a cualquier organización que tenga datos personales de residentes del estado de California a notificar a dichos residentes la sospecha de cualquier brecha digital que haya podido comprometer los datos, en caso de no estar cifrados. En la actualidad, existen varias regulaciones y leyes que tratan de regular este problema en distintas áreas de negocios, como Sarbanes Oxley (SOX), Payment Card Industry (PCI) Data Security Standard, Healthcare Services (HIPAA), Financial Services (GLBA), Data Accountability and Trust Act (DATA), etc.

En cada país encontraremos un ejemplo similar o leyes actualmente en preparación.

Pero ¿qué impacto puede tener un ataque exitoso contra una entidad?

Es difícil estimar el valor de los datos robados en sí, hay varias aproximaciones para dar un valor monetario a dichas pérdidas.

La siguiente tabla es un ejemplo:

Tipo de dato	Valor
Dirección	$0.50

Estimación del valor de los datos robados

Tipo de dato	Valor
Teléfono	$0.25
Teléfono no publicado	$17.50
Móvil	$10
Fecha de nacimiento	$2
Educación	$12
Historia crediticia	$9
Detalles de bancarrota	$26.50
Información judicial	$2.95
Historial laboral completo	$18
Archivo militar	$35
Tarjetas de crédito	$1-6
Identidad completa	$16-18
Disco duro con transferencias del Banco central de Rusia	$1800
Historial de llamadas a través de móviles	$110-200
30.000 e-mails	$5

Estimación del valor de los datos robados

Revisaremos los principales aspectos relacionados con la seguridad en las aplicaciones web y en las bases de datos que utilizan, tanto a nivel conceptual como a nivel más técnico, teniendo en cuenta las arquitecturas más populares hoy en día.

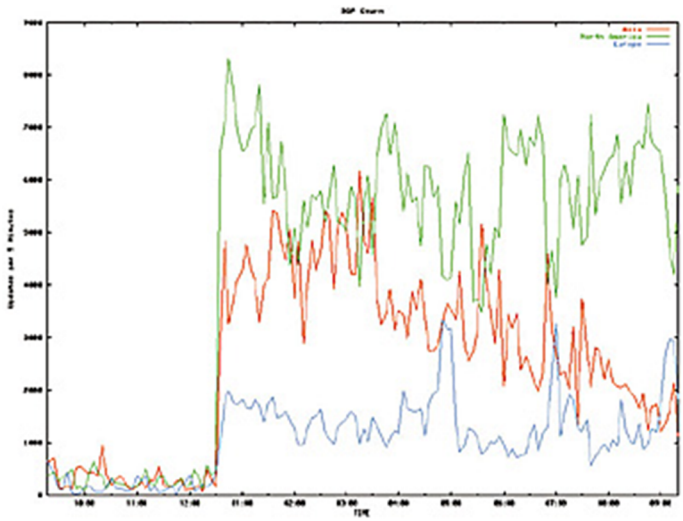
Detallaremos los problemas a los que nos enfrentamos para el diseño y defensa de un aplicativo web, incluyendo el punto de vista de un atacante, las técnicas que utilizará contra nosotros y las herramientas de las que dispone. Explicaremos cómo funcionan, cómo podemos protegernos, cómo minimizar riesgos, y cómo monitorizar la infraestructura para detectar cualquier problema.

2. Evolución de los ataques

El 25 de enero del 2003, un gusano conocido como *Slammer* infectó más de 75.000 máquinas en un lapso de unos 10 minutos, propagándose a una velocidad nunca vista hasta la fecha y causando denegación de servicio en algunos dominios y lentitud en el tráfico en general.

Este gusano explotaba una vulnerabilidad de desbordamiento de *buffer* en Microsoft SQL Server pública y con parche disponible desde 6 meses antes del ataque. Muchos administradores no habían parcheado sus sistemas de forma adecuada, pero sobre todo la difusión tan masiva fue debida a que muchos usuarios no eran conscientes de tener instalado *MS SQL Server Desktop Engine* (MSDE), el cual dispone de un motor de MS SQL Server.

Figura 1. Aumento del tráfico BGP, indicativo de atascos en Internet



Courtesy Akamai

Este ejemplo es ilustrativo en varios aspectos:

- En primer lugar, es un buen ejemplo de la amplísima difusión de las bases de datos, en este caso incluso sin el conocimiento de los usuarios de las mismas. El motor de la base de datos estaba embebido en otro producto de software.

- Una base de datos, a parte de cualquier diferencia conceptual que se pueda hacer, no deja de ser otro producto de software más, susceptible a ataques especialmente dirigidos para obtener los datos que alberga, o simplemente ataques genéricos que aprovechen una vulnerabilidad no corregida como en cualquier otro software.

- La gran presencia de bases de datos accesibles para un potencial atacante. Al estar conectadas a Internet, en ocasiones dando servicios, otras de for-

ma inconsciente para el usuario que la alberga, son un objetivo potencial. Aunque la distribución de este gusano fue indiscriminada, existen muchas técnicas para buscar bases de datos vulnerables a algún fallo conocido utilizando herramientas tan al alcance de todos como puede ser Google.

- La vulnerabilidad, no por ser conocida ni por disponer de un parche desde 6 meses antes del ataque, dejó de ser fatal. Esto pone de relevancia no sólo el problema de no disponer de una adecuada política de actualizaciones, sino del factor humano. Este factor, aunque difícilmente remediable, es en muchos casos determinante para explicar el porqué de muchos problemas de seguridad, por lo que la formación y la concienciación son un tema clave al mismo nivel que cualquier otro aspecto técnico.

En cualquier caso, se trata únicamente de un ejemplo de una vulnerabilidad explotada exitosamente por parte de un atacante. Hay decenas de vulnerabilidades, con distintas dificultades de explotación y con distinto grado de divulgación. La evolución que han seguido las vulnerabilidades y los ataques en bases de datos es similar a la evolución que han seguido en cualquier otro software según aumentaba en popularidad y en complejidad, con el atractivo que de ser explotado tiene una recompensa final muy jugosa. Pero también han crecido una serie de vulnerabilidades concretas para bases de datos, como pueden ser los ataques de inyección SQL y que se discuten más adelante.

Hay que contextualizar los problemas dentro del entorno en el que se han ido utilizando las bases de datos, ya que los diseñadores han ido dando respuestas a las necesidades del mercado con nuevas funcionalidades. En este contexto, normalmente han sido los atacantes los primeros en pensar nuevos vectores de ataque y los diseñadores han ido dando respuesta a los problemas según se iban detectando.

Sin embargo, en la actualidad se intenta pasar de un escenario reactivo a un escenario proactivo, en el que se detecten los posibles problemas antes de sacar un producto al mercado. Esto es debido a un cambio en la forma de pensar en las grandes compañías, pero sobre todo, en sus clientes. La seguridad ha pasado de ser un extra a ser una absoluta necesidad, y son los propios clientes los que la demandan, por lo que los creadores de motores de bases de datos y de software cada vez dedican más recursos para asegurar sus productos.

En la actualidad cualquier producto pasa varias etapas de calidad relacionadas con seguridad. En general se pasa una auditoría de código en la que se buscan posibles debilidades en el código fuente del aplicativo. También se realizan ataques de intrusión (*penetration test o pentest*) en el que un auditor juega el rol de un atacante e intenta con todos los medios a su disposición lograr el control del software auditado. De este modo, se buscan debilidades en las

que los diseñadores del software no han pensado, simulando el papel reactivo que han jugado las compañías pero en este caso antes de sacar el producto al mercado.

Aparte de intentar que el software tenga el mínimo de vulnerabilidades, es vital disponer de un servicio de respuesta rápida a incidentes, que permita publicar parches en respuesta a vulnerabilidades en el menor tiempo posible. Aun así, desde la aparición de una vulnerabilidad hasta la publicación e instalación del parche en el cliente, existe una ventana de exposición en el que el software es vulnerable al problema detectado. Y esto suponiendo que la vulnerabilidad detectada se reporte al vendedor y este publique un parche que lo solucione, ya que algunas vulnerabilidades no se reportan nunca al vendedor y son utilizadas por parte de atacantes: si alguien descubre una vulnerabilidad con la que acceder a un tipo de base de datos para sus propios fines, ¿por qué hacerla pública? Estas son las que se conocen como 0-day.

Incluso con la actual concienciación e inversión en seguridad, siguen existiendo gran cantidad de problemas.

> Durante el 2006 Oracle publicó cuatro actualizaciones críticas de seguridad relacionadas con servidores de bases de datos que resolvían más de veinte vulnerabilidades remotas. En el 2007, todavía hay más de cincuenta vulnerabilidades no parcheadas en Oracle Database Server.

Evidentemente, esto es únicamente un ejemplo, no significa ni mucho menos que sea el único motor de bases de datos afectado por este tipo de problemas. Esto significa que conseguir un sistema de bases de datos seguro al 100% no es posible, pero existen gran cantidad de medidas a implementar para minimizar la posibilidad y la gravedad de un eventual ataque.

Pero ¿qué tipos de ataques son los más usados? A continuación se listan algunos de los más habituales según OWASP:

- Ataques de fuerza bruta contra contraseñas
- Esnifar credenciales en la red
- Configuraciones inseguras por defecto
- Explotación de vulnerabilidades (públicas o no)
- Inyección SQL
- Uso de troyanos y *rootkits*
- Robo de dispositivos de almacenamiento físicos
- Personal interno

En realidad, la mayoría de intrusiones todavía se consiguen gracias a una administración deficiente. Aunque existen gran cantidad de métodos, siguen siendo los más triviales los que producen mejores resultados. Esto es debido a falta de concienciación o desconocimiento de los potenciales ataques. Por otra parte, hay gran cantidad de software heredado en las empresas que no se actualiza por miedo a las actualizaciones o por necesidades de otro software heredado que se utiliza en la actualidad. ¿Cuántas veces hemos oído “si funciona

no lo toques."? Esta situación es muy problemática, ya que en algunos casos las compañías dejan de dar soporte a ciertas versiones de software tras la publicación de versiones más recientes (como pasa por parte de Microsoft con Windows NT4).

Por lo tanto, aunque hoy en día el nivel de concienciación respecto a seguridad es mucho más alto que hace unos años, no es suficiente. Los ataques cada vez son más evolucionados y cada vez existen más herramientas para evitar los mismos, pero el factor humano sigue siendo el principal.

Es tarea tanto del administrador como de un auditor de seguridad implantar las medidas de seguridad disponibles para minimizar el riesgo de ataque y realizar un seguimiento mediante políticas de seguridad que aseguren la base de datos a lo largo del tiempo.

3. Perspectivas

Es difícil saber qué va a ocurrir en un futuro, pero en este apartado se hace un pequeño ejercicio al respecto considerando las últimas tendencias y las perspectivas en cuanto a seguridad para los próximos años.

3.1. Tipos de ataque

Lo que parece claro es que el número de ataques va a seguir creciendo. Durante los últimos años este aspecto se ha disparado y han aparecido nuevos vectores de ataque. Está claro que la difusión de Internet así como de los servicios proporcionados mediante dicho medio facilitan la posibilidad de buscar víctimas y de información para explotar vulnerabilidades. La figura del "atacante casual" o script-kiddie[1] es cada vez más frecuente.

(1)Usuario que, sin necesidad de tener conocimientos técnicos, usa un software de terceros o una vulnerabilidad conocida y fácilmente explotable para lograr una intrusión.

Es posible pensar en ello como alguien que lee cómo lograr una intrusión en ciertos tipos de sistemas de modo sencillo y utiliza este conocimiento para atacar un sistema vulnerable sin ningún propósito más que la curiosidad. Aunque este tipo de atacantes debería ser el menos preocupante por su falta de conocimientos técnicos, puede ser también muy peligroso precisamente por lo mismo: el atacante puede no ser consciente del riesgo que implica su acción precisamente por su falta de conocimientos técnicos. Una buena política de actualizaciones, una configuración correcta y el uso de medidas de seguridad adicionales deberían servir para que este tipo de atacantes no tuviesen éxito. El problema es que en caso de tenerlo, su falta de conocimientos puede provocar un daño mayor al que pretendía hacer precisamente por no saber qué está haciendo realmente.

Por otra parte, hay otro perfil de atacante muy capaz técnicamente y que estudia durante un tiempo a un posible objetivo. Existen en la actualidad personas que se dedican a la creación de software especializado contra un determinado objetivo a partir de un encargo. En general, este tipo de encargos se realizan mediante el uso de troyanos que afectan a los clientes de determinadas entidades o que explotan vulnerabilidades de los sistemas operativos para lograr acceder al entorno interno de la entidad y desde allí lograr credenciales para acceder a los sistemas críticos. Sin embargo, es posible que con el tiempo este tipo de ataques tengan como objetivo directo las bases de datos. En este caso, es posible que una política de actualizaciones sea importante para evitar vulnerabilidades relacionadas con versiones antiguas, pero no tiene por qué. Este escenario plantea el uso de una vulnerabilidad desconocida para el administrador, por lo que puede estar indefenso. Sin embargo, existen una serie de medidas que pueden paliar este problema, descubrirlo mientras se está produciendo o incluso evitarlo, como se explica durante este curso.

Los aplicativos web se utilizan mayoritariamente para la generación de páginas web, foros, sistemas sencillos de gestión, blogs, etc. Creados muchas veces por usuarios no profesionales que en general usan soluciones prediseñadas (con vulnerabilidades conocidas o no), suelen descuidar la actualización. Incluso son en ocasiones los propios creadores del software los que no proporcionan un servicio al respecto. Inevitablemente, van apareciendo vulnerabilidades asociadas a versiones antiguas, y una simple consulta en un buscador de Internet permite el acceso a miles de sistemas vulnerables. Este sistema se utiliza generalmente por parte de gusanos para su rápida propagación e infección masiva de modo casi instantáneo.

Por otra parte, los sistemas de bases de datos propiamente dichos, con dedicación exclusiva y debidamente actualizados, tienen un problema, ya que cada vez son más grandes. La gran cantidad de servicios demandados por los usuarios reflejan una necesidad del mercado a la que los fabricantes tienen que dar respuesta, pero el aumento de complejidad también supone la aparición de nuevas posibilidades de ataque.

Hay que buscar siempre el equilibrio entre funcionalidad, necesidad y riesgo. La política tradicional en este aspecto es evitar instalar cualquier servicio innecesario.

Pero no todo son malas noticias. Los fabricantes cada vez implementan más servicios dedicados a seguridad. Aunque algunos fabricantes ya los implementan en algunos de sus productos en la actualidad, es previsible su implantación masiva. El primer servicio es el de cifrado de datos de forma transparente. Aunque puede suponer una pérdida de rendimiento en ciertos entornos críticos, en otros es muy importante tener todos los datos cifrados de forma automática, ya que facilita la administración y evita errores humanos. Por otra parte, casi todos los sistemas funcionan ya con comunicaciones también cifradas y que evitan la posibilidad de que un atacante acceda a los datos transmitidos entre el cliente y el servidor. Es de esperar que estos sistemas se utilicen de forma transparente y que se evite cualquier otro tipo de comunicación en texto plano para una mayor seguridad del sistema.

3.2. Modelos de programación

En cuanto a modelos de programación que interactúan con bases de datos, también han seguido una evolución. La utilización de cierto tipo de ataques, como la inyección SQL, ha supuesto un cambio en el paradigma de programación utilizado hasta el momento. Los nuevos *frameworks* de creación de software que interactúa con bases de datos ya incorporan medidas para evitar este tipo de problemas. En este marco, la situación es parecida a la del uso masivo de motores de bases de datos por parte de software de terceros, aunque ahora el problema no es del motor sino del software que interactúa con la misma.

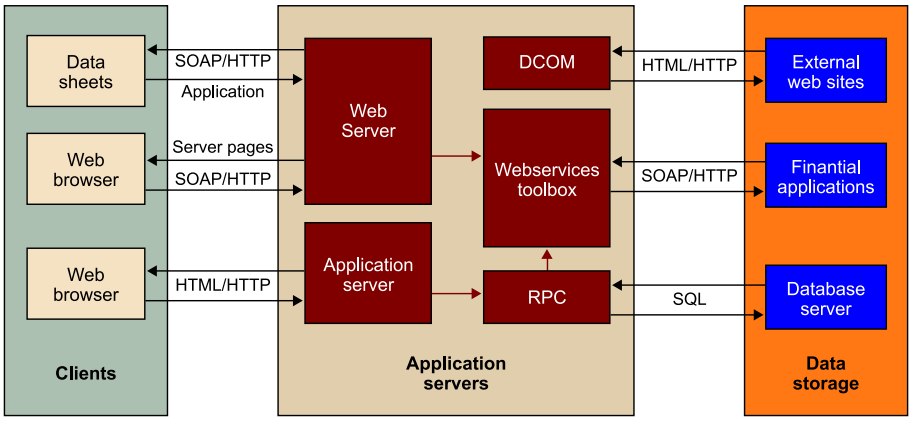
La prevención en este escenario pasa por el uso de las medidas existentes en el momento de la creación del software, y por el uso de código bien estructurado que permita la adopción sencilla de modificaciones que prevengan nuevos ataques. También se ha popularizado el uso de filtros, IDS e IPS (*intrusion detection system, intrusion prevention system*). Aunque no es previsible la adopción de estas medidas en la parte del motor de la base de datos, es importante siempre tener en cuenta el entorno en el que se encuentra la misma y utilizar las medidas de protección en todos los niveles. Es un error pensar que se puede tener una base de datos perfectamente securizada sin pensar en todo su entorno y en tenerlo también debidamente protegido.

4. Arquitectura de aplicaciones web

4.1. Arquitectura genérica

La arquitectura de los sistemas web establece las directrices que se deben seguir para el diseño de los sistemas software. La figura 2 muestra la típica estructura en tres capas para esta arquitectura y el movimiento de datos entre las interfaces de usuario y las capas de aplicación y los almacenes de datos.

Figura 2. Arquitectura web general basada en tres fases



Esta arquitectura en tres capas permite el desarrollo modular de las interfaces de usuario dependientes de un dispositivo específico (un programa determinado) para su comunicación a través de una estructura de envío de servicios web (*web services*) multicanal. Por otro lado también permite incorporar procesos o estrategias de negocio existentes o totalmente nuevas.

El envío de *web services* está ajustado al modelo de paso de mensajes descrito en los protocolos "*OASIS web services reliable messaging (WSRM)*" propuesto por IBM, BEA, TIBCO y Microsoft (acorde con W3C).

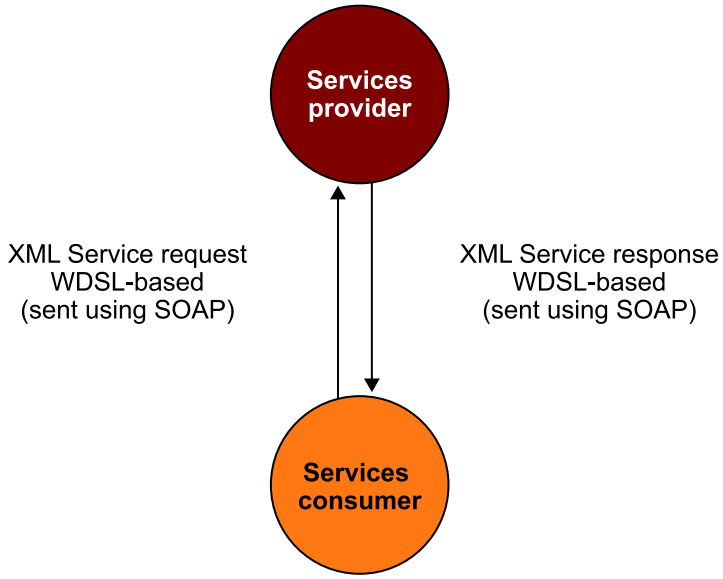
- En el lado del cliente, el navegador web proporciona la interfaz de usuario y la lógica de presentación.

- En el lado del servidor, el servidor web captura todas las peticiones HTTP enviadas desde el usuario y las propaga al servidor de aplicación el cual implementa la lógica de la aplicación.

- En el lado del almacén de datos, se procesan de forma transicional e histórica los datos de las operaciones diarias y se almacenan en la base de datos del sistema por el gestor de bases de datos.

El servidor de aplicación envía una consulta (*query*) al servidor de bases de datos y obtiene el conjunto de resultados. El servidor prepara la respuesta desde una serie de procesos y la envía al servidor web para que este la haga llegar al cliente. Un usuario registrado puede iniciar sesión obteniendo un perfil en el que se especifican sus privilegios, derechos de acceso y el nivel de complejidad. Una arquitectura basada en esta estructura trifásica puede dar soporte tanto a una intranet como a un entorno accesible desde Internet. El usuario puede utilizar, por ejemplo, un navegador web para acceder al servidor de páginas web usando HTML y HTTP. El núcleo del sistema está situado en el lado de los servidores de aplicación y proporcionará un conjunto de *web services*. Estos servicios se pueden comunicar unos con otros. Esta comunicación puede implicar un simple traspaso de datos o puede implicar el intercambio de información necesario para que dos o más servicios trabajen de forma conjunta para realizar una única actividad.

Figura 3. Arquitectura Orientada a Servicios Básica



La figura 3 ilustra la arquitectura orientada a servicios básica (*services oriented architecture, SOA*). Se muestra un consumidor de servicios enviando una petición (mensaje de *request*) al proveedor de servicios. Este último devuelve un mensaje de respuesta al consumidor (mensaje de *response*). Tanto el *request* como las subsiguientes conexiones de respuesta están definidos de tal manera que son decodificables por los dos agentes participantes en la comunicación. Para completar este esquema tan sencillo es necesario remarcar que un proveedor de servicios puede también actuar como un consumidor.

En el ejemplo que está ilustrado en la figura 3 todos los mensajes están formateados usando SOAP, que puede usar distintos protocolos (HTTP, SMTP, etc.). SOAP proporciona el marco para enviar los mensajes de los *web services* a través Internet o de la intranet. Este marco consta de dos partes:

- Una **cabecera** opcional en la que se puede indicar información sobre la autenticación o la codificación de los datos.

- Un **cuerpo** que contiene el mensaje. Estos mensajes pueden definirse usando la especificación WSDL que emplea XML para definir los mensajes. XML es un lenguaje de etiquetado que puede ser utilizado tanto por el proveedor de servicios como por el consumidor para el intercambio de información sin tener que ajustarse a un protocolo en el que el orden de los datos sea muy estricto.

La presencia de los estrictos límites entre cada una de las capas de la figura 2 se trasladará a las aplicaciones software implementadas sobre esta arquitectura. Una distribución equilibrada de los componentes de cada capa entre las computadoras de la red garantiza un nivel de flexibilidad que no se alcanza con otras arquitecturas alternativas. Como resultado, los recursos computacionales alcanzan un alto rendimiento que se ve reflejado en la alta complejidad de la funcionalidad que se puede implementar.

Resumen

Las características que definen esta arquitectura son las siguientes:

La información final es solicitada por el cliente y se genera en la capa de servidor. Con ello se prescinde de que sea el cliente el último responsable del procesado de los datos.

Todos los recursos de información y la lógica de la aplicación están concentrados en el servidor. En contra de la disposición distribuida que se proponía en arquitecturas ahora obsoletas (por ejemplo, el modelo cliente-servidor).

Los protocolos que se utilizan para el intercambio de datos entre la capa del cliente y el resto de las capas son estándares, como TCP/IP sobre Internet.

También se estandariza el control centralizado no solo del servidor, sino también de los equipos de los clientes, al menos en términos de software. De forma que en cada estación de trabajo es suficiente con tener un programa de navegación estándar.

Estas *workstation* pueden ejecutar programas locales y, de forma remota, los de otros equipos de la red.

Todas estas características, a excepción de la última, ayudan a mitigar el problema de seguridad informática. La concentración de todos los recursos de información y aplicación en capas ajenas a la que alberga a los clientes simplifican el diseño y la administración del sistema gestor de la seguridad de la información. Además, el diseño de los sistemas de protección de los objetos localizados en un único lugar es más sencillo que si esta localización fuese distribuida.

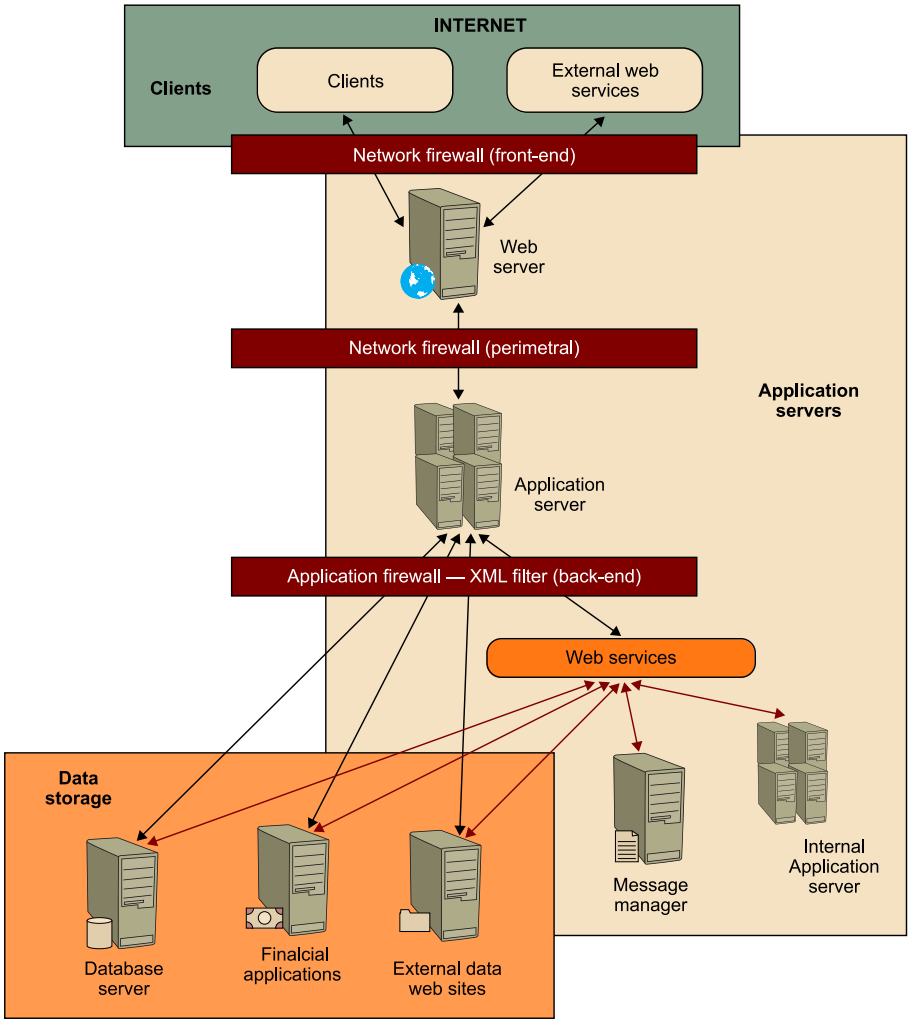
El uso de estándares abiertos como TCP/IP para el intercambio de datos entre los equipos de la red garantiza la unificación de todos los métodos de interacción entre *workstations* y servidores. Gracias a ello es posible proponer una solución para la securización de los intercambios de información válida para todos los sistemas. El control centralizado ejercido desde los servidores sobre los equipos cliente reduce la probabilidad de errores cometidos por descuidos o falta de formación de operarios o usuarios. Estos errores son una de las principales amenazas del sistema y son potencialmente muy peligrosos.

Como se apuntó anteriormente, en esta arquitectura, el procesamiento de la información distribuida asume que los programas obtenidos desde los servidores van a ser ejecutados en las *workstations*. Este sistema de procesamiento distribuido permite la concentración de toda la lógica de la aplicación en la capa de servidor. Sin embargo, la posibilidad de ejecutar programas desde el servidor en un cliente genera nueva amenazas que obliga a ser más exigente con los sistemas de protección.

Como se mencionó anteriormente, la implementación más común de *web services* utilizan *SOAP*. Esto constituye un potencial problema ya que *SOAP* usa el protocolo HTTP. Y esto implica que si no se establecen medidas de protección precisas es posible llegar, usando HTTP, hasta los sistemas de almacenamiento de la figura 2; lo que puede suponer un riesgo de seguridad significativo.

La forma más habitual de solucionar este problema es integrar la arquitectura web con una arquitectura de comunicaciones segura basada en una configuración eficaz de *firewalls*. La figura 4 resume las principales características de esta arquitectura general securizada.

Figura 4. Estructura segura basada en redes perimetrales para la arquitectura web de tres capas



Como se aprecia en la figura, esta arquitectura incorpora un *firewall* de aplicación basado en XML sobre los *web services* internos para filtrar los mensajes que les llegan desde el servidor de aplicación.

4.2. Tipos de aplicaciones

A partir de la arquitectura general propuesta en el apartado anterior, es posible establecer una clasificación de aplicaciones que permite distinguir entre tres categorías:

- *Web site* público
- Intranet
- Extranet

Las detallaremos a continuación.

1) *Web site* público, Internet o *World Wide Web*

En este escenario la organización dispone de los recursos para montar un sistema de difusión de la información sin restricciones. No existen restricciones aplicables a los usuarios que motiven la habilitación de sistemas arquitecturales para el control de accesos. Todos los mecanismos de autenticación dependerán de la lógica de la aplicación y serán, por tanto, gestionados por el servidor de aplicación.

Las aplicaciones típicas son aquellas en la que no se establece un sistema interactivo de negocio sino puramente informativo.

2) Intranet

> En el sentido más amplio del término, una intranet es una red de computadores que establecen sus interconexiones usando protocolos desarrollados para Internet, pero que no está integrada en la *World Wide Web*.
>
> Sin embargo, desde una perspectiva puramente práctica, esta definición suele complementarse con la afirmación de que una intranet está formada por aquellos sitios web o aplicaciones que pueden encontrarse en la red y que pueden usarse como recursos y plataformas de comunicaciones únicamente para la organización que las gestiona; no la red de comunicaciones en sí misma sino los recursos que serán solo accesibles de forma privada.

En un entorno empresarial en el que constantemente se está incrementando la complejidad de las infraestructuras, las organizaciones tienen que afrontar el desafío que supone adaptar estas infraestructuras a las nuevas tecnologías

de captura y compartición de información con el objetivo de alcanzar una alta efectividad en la toma de decisiones. Existen tres categorías dentro de la tecnología de la información que necesitan de un entorno basado en intranet para alcanzar este objetivo:

- La planificación de recursos de empresa a nivel transaccional (*enterprise resource planning*, ERP).
- La administración de la relación con los clientes (*customer relationship management*, CRM).
- El sistema de control de la producción, a nivel de tiempo real (*manufacturing execution system*, MES).
- La aplicación de estas tres tecnologías sobre un escenario como una intranet persigue el cumplimento de los principios básicos, como son el aumento de la producción, alcanzar el *zero downtime* y reducir los costes.

3) Extranet

Una extranet mantiene una posición intermedia entre intranet e Internet. En este escenario surgen cuando el uso de los recursos privados de una intranet se extiende para que sean accesibles por usuarios externos a la organización, típicamente los clientes de la misma.

Las extranets se mantienen desde la red interna de la organización, garantizando a los usuarios el acceso a los recursos, pero utilizan Internet para el envío de información siempre y cuando algunos de sus usuarios son externos a la red interna de la organización. Es decir, en una extranet se mantienen recursos privados para usuarios que pueden establecer un acceso externo a la organización.

En realidad la necesidad de adaptar el acceso a los recursos respecto a los que se planteaba en el caso de la intranet no se debe a una alteración en los objetivos que se persiguen. Estos objetivos seguirán siendo los mismos. Sin embargo, lo que justifica la utilización de una extranet es la necesidad de aplicar mayor flexibilidad a la relación con los clientes. De hecho, en una extranet es posible gestionar dos modelos de negocio imposibles en una intranet: el comercio electrónico de empresa a empresa (*business to business*, B2B) y el comercio electrónico entre empresa y consumidor (*business to consumer,* B2C).

4.3. Amenazas a la arquitectura web

4.3.1. Estado de la seguridad web

Ganando algo de perspectiva en el cumplimiento de los principios básicos de productividad y reducción de costes asociados a la gestión de las aplicaciones web, se puede concluir que es necesario abordar un análisis del estado de la seguridad de estos sistemas. Para ello es imprescindible emprender un intento de localizar los puntos críticos de las aplicaciones antes comentadas, con el fin de evaluar qué amenazas son las que se ciernen sobre el sistema. Para ello se deben evaluar datos estadísticos sobre ataques implementados sobre diferentes sitios web. A partir de estos datos evaluar a qué tipo de amenazas corresponden.

La localización de fuentes a partir de las cuales extraer datos de valor estadístico es un problema de difícil solución. Las implicaciones que tiene sobre la imagen de la empresa y las pérdidas derivadas del deterioro de las mismas hacen que una política frecuente de las empresas sea la ocultación de estos incidentes de seguridad en un intento de minimizar los efectos sobre su productividad y la relación con sus clientes. Por ello no es posible disponer de la suficiente cantidad de datos oficialmente reconocidos por las empresas para establecer resultados con validez estadística.

La única solución para obtener información sobre qué tipo de sistemas son más frecuentemente víctimas de ataques, cuáles son las motivaciones de los atacantes o si el ataque está enfocado al sistema operativo o a las aplicaciones, es preciso localizar fuentes alternativas de datos. De hecho, una alternativa para esta búsqueda de información es utilizar sitios web como www.zone-h.org en los que es posible obtener datos sobre ataques implementados por distintas asociaciones de *hacker* y sobre sus motivaciones.

Atendiendo a este repositorio de información, y teniendo en cuenta que no se trata de una información absolutamente fiable, sí que es posible extraer una serie de conclusiones muy interesantes:

El número de intrusiones web se ha ido decrementando año tras año debido a la implantación de nuevas contramedidas y a las actuales arquitecturas web. Esto se puede deber a que la implementación de los ataques cada vez exige una mayor profesionalización para contrarrestar los altos niveles técnicos introducidos con las últimas tecnologías de protección. De hecho, durante los tres primeros meses del 2009 el número de ataques registrados por *zone-h* ha sido de 54.960 frente a los 108.514 ataques registrados en el primer cuatrimestre del 2008.

Los sistemas atacados no mantienen un perfil claramente definido. De hecho se puede concluir que se atacan todas las tecnologías, desde los sistemas operativos a las aplicaciones, sin atender a fabricantes específicos.

Los motivos son variados pero se pueden resumir en los siguientes:

- económicos,
- por venganza,
- políticos,
- diversión.

A partir de estos resultados se puede concluir que, a pesar de que parece que se está avanzando en la dirección adecuada en cuanto a la implementación de medidas de protección, todavía se están asumiendo muchos riesgos al plantear las soluciones web. Es preciso reducir los más de 50.000 ataques exitosos por trimestre. Para ello, es preciso profundizar más en el análisis de las amenazas definidas sobre la arquitectura y las aplicaciones web.

4.3.2. Evaluación de riesgos sobre la arquitectura web

La evaluación de los riesgos que surgen a partir de la propuesta de arquitectura web es tan compleja que es preciso descomponer el análisis para determinar sobre qué componentes de la arquitectura se ciernen las amenazas potencialmente más peligrosas.

1) Riesgos sobre los clientes

En la mayoría de los casos los clientes acceden a los recursos de forma local o remota a través del navegador de Internet. Este software ejecuta código a nivel de usuario, con el nivel de privilegios que este tenga definido en su perfil. Los códigos que se ejecutan desde el navegador son potencialmente peligrosos debido a la gran funcionalidad que se puede implementar con ellos. Los lenguajes más extendidos son HTML/DHTML, vbScript, JavaScript o JScript. El navegador web permite incrustar programas realizados como applets de Java, ActiveX o Shockwave de Flash. Además normalmente presenta el código sin establecer ninguna medida de protección; no suelen aparecer cifrados ni ofuscados. Esta última característica es especialmente sangrante cuando existen múltiples alternativas para establecer un nivel de protección en el código (por ejemplo, Atrise Stealth o HTMLLock).

En realidad ninguna protección en cliente es buena, ya que además de los ataques directos al navegador web del cliente directos o con decompiladores, existen las técnicas de *man in the middle* (MiM) que apartan totalmente al usuario de la posibilidad de evitar el éxito de un ataque lanzado sobre el sistema. Algunas herramientas que permiten evaluar la potencial peligrosidad de los ataques MiM son las conocidas Achilles, BurpSuite u Odysseus.

2) Riesgos sobre la lógica de la aplicación

El servidor de la aplicación concentra toda la lógica de la aplicación y para ello necesita ejecutar código en contextos privilegiados, convirtiéndose con ello en un objetivo formidable para un ataque. Estos códigos están desarrollados usando potentes lenguajes de programación que incrementan mucho la criticidad de un posible ataque sobre el sistema. Como se ha explicado, este componente accede a la base de datos, envía programas a clientes, transfiere ficheros y ejecuta comandos sobre el sistema. Además, proporciona soporte a herramientas para administración de otras aplicaciones y presenta códigos de ejemplo para facilitar las tareas de configuración.

Con todo ello este componente es el candidato ideal para convertirse en el objetivo de los ataques o, en la mayoría de los casos, servir de vehículo para la implementación de ataques sobre otros componentes (sobre los clientes a través de ataques de *cross site scripting*, sobre los almacenes de datos a través de ataques de inyección de código (*SQL injection, LDAP injection o XPATH injection*; sobre el sistema con la inyección de ficheros).

3) Riesgos sobre los almacenes de datos

El almacenamiento de información suele considerarse la clave de negocio, ya que un ataque a su confidencialidad o su integridad puede acarrear enormes pérdidas económicas así como, en determinados casos, la implicación en el quebrantamiento de la Ley de Ordenación y Protección de Datos (LOPD). El almacenamiento de los datos suele realizarse en bases de datos relacionales o jerárquicas que serán accedidas mediante lenguajes de mayor o menor potencia. En el caso de las bases de datos basadas en objetos los lenguajes son muy avanzados (3.ª o 4.ª generación) y permiten una altísima interacción con el sistema. Un ataque exitoso sobre este tipo de sistemas puede resultar altamente crítico, dada la facilidad en su explotación una vez que se hayan ganado los privilegios necesarios. De hecho, un ataque de este tipo podría resultar demoledor si se ha configurado el sistema para permitir la ejecución de programas sobre el sistema.

En el caso de las bases de datos jerárquicas, la información almacenada es igualmente importante. Es frecuente almacenar el catálogo global de datos y proporcionar un lenguaje para garantizar el acceso. Normalmente, estos lenguajes son más limitados que los definidos para otros entornos, pero aun así, si se ha ganado un nivel de privilegios suficiente podrán usarse para implementar un ataque que permita la extracción de toda la información almacenada.

5. Arquitectura de bases de datos

Aunque es muy difícil conocer todas las arquitecturas en profundidad, es muy necesario conocer los principios de todas ellas. Un estudio general permite establecer paralelismos entre los principales productos y la evolución que han seguido los mismos a lo largo del tiempo. Además de contemplar el escenario en el que se crea una nueva base de datos y se compra una licencia de uso de cualquier fabricante de su última versión de software, es muy común trabajar con versiones antiguas, de ahí la importancia de conocer las distintas versiones de cada arquitectura.

Existe un principio que formula que la base de datos más segura es la que mejor se conoce. La elección debe ser un compromiso entre necesidad y funcionalidad que ofrece cada arquitectura, y una vez hecha la elección entre la oferta del mercado, estudiar en profundidad la misma para lograr un nivel óptimo de seguridad.

5.1. Oracle

Probablemente, Oracle es la base de datos más conocida y popular del mundo. Una muestra de su popularidad es que se encuentra presente en 98 de las 100 industrias incluidas en el top 100 de *Fortune*. Fue uno de los primeros vendedores del mercado y se ejecuta en una gran cantidad de plataformas, amén de proporcionar un muy buen producto, lo que explica su popularidad.

Sin embargo, ha tenido un historial de problemas relacionado con la gran cantidad de funcionalidades que proporciona. En especial, inyección de PL/SQL (el lenguaje de *scripting* proporcionado por el producto) y desbordamientos de búfer en distintas partes de código han sido algunos de los principales problemas históricos de este producto.

5.1.1. Historia y evolución

Dado que Oracle es un producto muy veterano en el mercado, existe una larga historia de versiones del mismo. Es más, cuando se habla de Oracle, se habla de un término ambiguo que puede dar lugar a confusión debido a la gran cantidad de productos que existen de este vendedor.

Las primeras versiones tienen interés a nivel histórico, aunque hoy en día ya no están vigentes en el mercado. Las versiones más “actuales” a día de hoy son:

- Oracle versión 8i (1999): Incorpora mejoras para responder a las necesidades de Internet (como la "i" en el nombre indica). Incluye una máquina virtual de Java en el motor de la base de datos (JVM).

- Oracle 9i (2001): Soporte para XML y *Clustering*, entre 400 nuevas funcionalidades.

- Oracle 10g (2003): En este caso, la "g" del nombre pone énfasis en " *grid computing*".

- Oracle 10.2.0.1 (2005): Conocido como Oracle 10g Release 2 (10gR2).

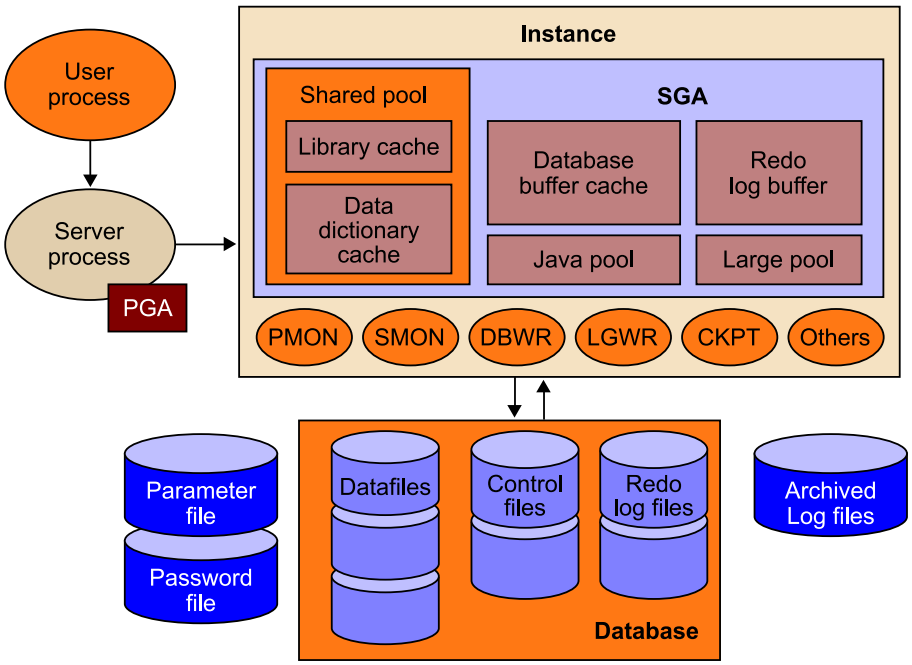
Las distintas versiones se proporcionan en diferentes productos o ediciones: Enterprise Edition, Standard Edition y Standard Edition One, además de la versión Express para evaluación y entornos de desarrollo. Todas utilizan la misma arquitectura, y aunque proporcionan distintas funcionalidades, a nivel conceptual y en cuanto a seguridad tienen las mismas características.

Aparte del motor de base de datos, existe gran cantidad de software que proporciona funcionalidad adicional e interacción con distintos aspectos de la base de datos, como desarrollo de software (Oracle Developer Suite), un servidor de aplicativos (Oracle Application Server), *clustering*, etc.

5.1.2. Arquitectura

Aunque la arquitectura específica para cada una de las versiones de Oracle es distinta, existen una serie de conceptos que constituyen el esqueleto de una base de datos Oracle heredados desde la versión 7 hasta las más recientes. No es el objetivo de este material conocer con detalle todos los aspectos de la misma, pero sí las ideas principales:

Figura 5. Arquitectura de Oracle



En esta imagen se observan los participantes en una base de datos Oracle.

- **Proceso de usuario**: Se conecta al servidor para interactuar con la base de datos.
- **Proceso del servidor**: Escucha los procesos de usuario y realiza las peticiones a una instancia de la base de datos.
- **PGA** (***program global area***): Memoria reservada para un proceso de usuario. Se libera al morir la sesión.
- **Instancia**: Conjunto de procesos tanto lógicos como de control para acceder a los datos en sí, almacenados en ficheros físicos. Accede a una y sólo una base de datos. Dentro de la instancia hay una serie de subprocesos que se detallan a continuación:
  - SMON (*system monitor*): Recupera la instancia al arrancar la base de datos.
  - PMON (*process monitor*): Monitoriza los procesos de usuario y libera los recursos que ocupan los mismos cuando acaban.
  - DBWR (*database writer*): Escribe los datos físicamente en los ficheros de *logs* de redo cuando es necesario liberar dichos *buffers* de la instancia.
  - ARCH (*archiver*) (antiguo LGWR): Una vez los archivos de *logs* de redo están llenos, escribe los datos en los archivos correspondientes para liberar espacio.
  - CKPT (*checkpoint process*): Cada vez que se vacían los *buffers* y se escriben en ficheros de *logs*, se produce un *checkpoint*. Este proceso es el encargado de coordinar todas las operaciones correspondientes a dicho evento y guardar la coherencia de la base de datos mediante la escritura de los datos en todos sus archivos correspondientes y asegurando que las operaciones han tenido éxito.

– RECO (*recover*): Elimina y limpia cualquier dato incoherente resultante de una transacción distribuida fallida. Este proceso se incluye en la versión de Oracle 10g.

Hay que tener en cuenta que los principales procesos se mantienen a través de las distintas versiones. Sin embargo, es inevitable que algunos vayan cambiando a raíz de las nuevas funcionalidades que se incluyen con las nuevas versiones, como se aprecia en el último de los procesos anteriormente descritos.

- SGA (*system global area*): Es el área de memoria creada al arrancar una instancia de Oracle. Tiene una serie de subáreas:
  – *Shared pool*: Almacena tanto la estructura e índices de los últimos objetos a los que se ha accedido en la base de datos como de las funciones y procedimientos PL/SQL utilizados.
  – *Redo log buffer*: Utilizado para deshacer las acciones sobre los datos físicos en caso de realizar un *rollback* de las acciones realizadas durante una sesión.
  – *Database buffer cache*: Hace referencia a todos los datos que cachea el servidor para evitar acceder a los datos físicos cada vez que se realiza una consulta, favoreciendo el rendimiento para consultas repetitivas que hacen referencia a los mismos datos.
  – *Java pool y large pool*: Conceptualmente son lo mismo que el *sharedpool* , sólo que se dividen para utilizarlos tanto en acceso de entrada/salida como para requerimientos de parseo de Java. No tienen un especial interés.

- *Database*: Hace referencia a los archivos físicos que almacenan los datos. Se dividen en:
  – **Archivos de control** (extensión .ctl) que almacenan información acerca de los archivos que forman parte de la base de datos y se encargan de mantener la consistencia de la misma.
  – **De datos** (extensión .dbf) que almacenan físicamente los datos (hay que tener en cuenta que una única tabla puede almacenarse en N ficheros).
  – **De redo-logs** (extensiones .rdo y .arc) para deshacer acciones en los datos físicos hasta un punto de control, especialmente pensado para recuperar el sistema a un punto seguro en caso de un fallo.

- *Parameter file, password file y archived log files*: Archivos de control que guardan los parámetros específicos para un servidor Oracle (normalmente init.ora), contraseñas (en desuso en versiones actuales del servidor) y archivos de log del servidor.

Hasta aquí se ha comentado la arquitectura básica a nivel de procesos y de archivos, es decir, la arquitectura física. Sin embargo, la estructura interna lógica a nivel de tablas y de estructuras de datos es incluso más importante, ya que normalmente se trata el nivel de seguridad dentro del contexto de un usuario con acceso a la base de datos.

Los datos en Oracle se organizan por bases de datos, o *tablespaces*. Para cada una de estas bases de datos, los nombres son únicos. En cada instalación de Oracle existe una base de datos por defecto en la que se guarda el esquema (*schema*) de todos los objetos del sistema. Esto es, todos los usuarios, bases de datos, tablas, vistas, procedimientos, funciones, etc. Como es normal, no todos los usuarios tienen permisos para acceder a esta base de datos y debe estar bien protegida, ya que contiene información vital, en este caso, metadatos.

Dicha base de datos es *sys*, que se corresponde con el catálogo del sistema. A continuación se listan algunos de sus objetos más importantes:

- `all_tables` : Contiene información de todas las tablas del sistema.
- `dba_users` : Contiene información de usuarios que tienen permisos de DBA.
- `all_users` : Contiene información de todos los usuarios de la base de datos.
- `tabs` : Contiene información de todas las tablas de usuario.
- `user_tab_columns` : Contiene información de columnas de las tablas de usuario.
- `all_db_links` : Contiene información de los enlaces con otras bases de datos.

La diferencia entre muchas de estas estructuras es el prefijo, que especifica qué usuario tiene permisos para consultarlas y qué nivel de información ofrecen. Las que tienen prefijo *dba_* solo son accesibles por usuarios con dicho nivel de permisos y son las que ofrecen más información. Hay que tener en cuenta que para no tener datos redundantes, estas estructuras son vistas, por lo que los datos reales no se almacenan en las mismas.

Por eso en algunos casos pueden modificarse para hacer invisible información para algunos (o todos) los usuarios.

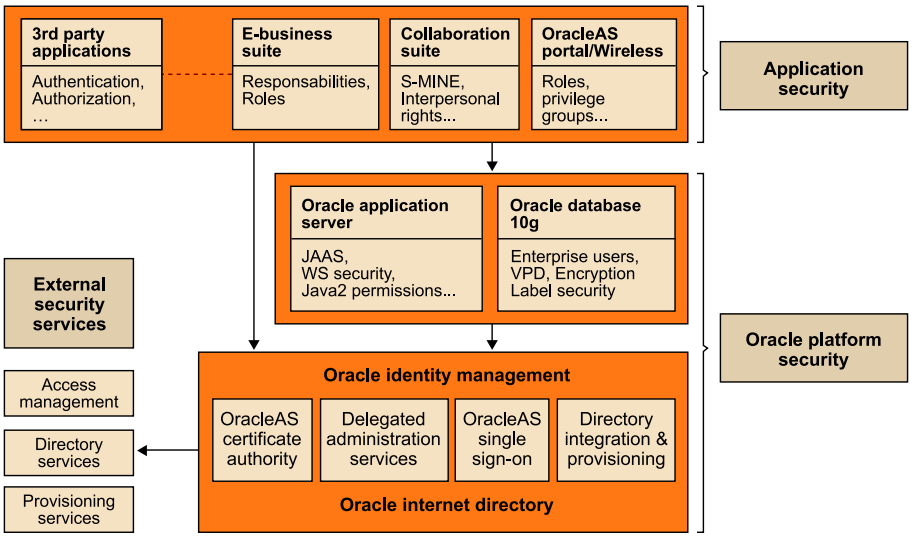
La tabla llamada *dual* se utiliza como comodín por Oracle para hacer consultas sin tener como objetivo ninguna tabla en concreto. Por ejemplo, se puede seleccionar el usuario de la sesión de la base de datos con la consulta:

```
select user() from dual
```

En este caso no se está haciendo una consulta de una información residente en la tabla *dual*, sino que se utiliza para usar la sintaxis SQL teniendo en cuenta que dicha tabla en realidad solo contiene una columna llamada *dummy* y un único registro.

En cuanto a la autenticación contra la base de datos, Oracle permite autenticar mediante su propio sistema, mediante el sistema operativo o mediante un sistema mixto. A partir de la versión 10g, Oracle implementa el *Oracle identity management*, que permite la integración de la autenticación de la base de datos con el sistema operativo, con sistemas LDAP, sistemas de PKI, etc.

Figura 6. *Oracle identity management*



Una vez autenticados, los usuarios tienen un rol en la base de datos que le otorga una serie de permisos. Este apartado es más interesante que el medio de autenticación, ya que en función del rol que tenga asignado el usuario tendrá unos permisos u otros.

Los permisos para estos roles se almacenan en las tablas:

- `system_privilege_map`
- `table_privilege_map`

Aunque a partir de la consulta del objeto `dba_users_privs` se pueden consultar los permisos de un usuario.

Los roles con máximos permisos en el sistema son SYS y SYSTEM. Existen varias cuentas con privilegios de DBA (*data base administator*), que sería la segunda cuenta con más permisos del sistema, como *ctxsys, wksys, mdsys* o *sysman*, aunque existen listas con más de 400 cuentas utilizadas por distintas versiones de Oracle. Este es uno de los problemas de seguridad en las instalaciones de Oracle por defecto.

A continuación se comentan los procesos que ejecuta una base de datos Oracle, así como la función de cada uno de ellos y se destacan los más importantes.

En primer lugar, hay que tener en cuenta que Oracle puede ejecutarse en una gran cantidad de sistemas operativos, lo que sumado a las distintas versiones de la base de datos resulta en un número potencialmente alto de combinaciones. Sin embargo, se puede usar el siguiente criterio como guía.

En sistemas Windows, el proceso principal de Oracle corre bajo el nombre de *oracle.exe*.

En sistemas *nix (Unix y Linux), existe un ejecutable para cada una de las funciones que se destacan anteriormente en este apartado. Este conjunto de funciones depende de la versión exacta de la base de datos, pero en general se encuentran los procesos siguientes:

- `pmon`
- `smon`
- `reco`
- `dbwr`
- `lgwr`
- `ckpt`

Uno de los procesos más destacables es el *TNS listener* (*transparent network substrate*). Este proceso juega el rol de "proceso del servidor" descrito anteriormente, es decir, espera las peticiones de un usuario y las transmite a la base de datos para retornar la respuesta de la misma, es decir, atiende las peticiones de red.

En este caso hay dos binarios asociados: el *tnslnsr*, que es el *listener* mismo, y el *lsnctrl*, que es la utilidad de control del *listener*.

En general, este proceso escucha en el puerto 1521 TCP, aunque es un parámetro configurable en cada instancia de la base de datos. Cuando un cliente quiere conectar con la base de datos, realiza una petición al *listener*, el cual conoce el estado actual de la base de datos. En caso de estar todo correcto, responde con un puerto al que el cliente debe conectar para realizar sus consultas directamente contra la base de datos, mediante la autenticación correspondiente.

Este proceso puede estar protegido o no. En las versiones anteriores a 10g no lo estaba en la configuración por defecto, lo que suponía un serio riesgo de seguridad, ya que se permitía la administración de *lsnctrl* remotamente a no ser que el administrador de la base de datos lo evitara con una configuración personalizada.

En caso de encontrar un proceso *TNS listener* desprotegido, es posible conocer información acerca de la instancia de Oracle, como conocer los nombres de las bases de datos que alberga y más información del sistema. Esto sin tener en cuenta un gran número de vulnerabilidades más avanzadas que se describen más adelante. En caso de estar protegido, también es posible obtener información acerca de los nombres de las bases de datos que alberga mediante ataques de fuerza bruta.

Ved también

En el módulo "Ataques a aplicaciones web" veremos más detalladamente los ataques de fuerza bruta.

Otro proceso destacable es el *Oracle intelligent agent*, que escucha en los puertos TCP 1748, 1808 y 1809. Realiza varias funciones, entre las cuales guardar datos relativos a la administración de los datos y a su rendimiento. Es interesante el proceso porque se puede consultar de forma remota sin necesidad de presentar ningún tipo de autenticación, de forma que proporciona información que puede resultar interesante a un atacante, como los procesos en ejecución, el uso de memoria, etc.

5.2. Microsoft SQL

Microsoft SQL Server es una base de datos muy popular, en buena parte gracias a la gran cantidad de sistemas Microsoft Windows en todo el mundo, lo que no quiere decir que no sea un buen producto. Este motor de base de datos únicamente está disponible para sistemas Windows.

Microsoft comenzó la evolución en solitario de su motor de base de datos basado en el código desarrollado por Sybase, colaborador conjunto en las primeras versiones de SQL Server. Sin embargo a finales de los noventa, Microsoft adquirió todos los derechos de SQL Server y comenzó su desarrollo basado en el código de Sybase, pero reescribiendo su totalidad.

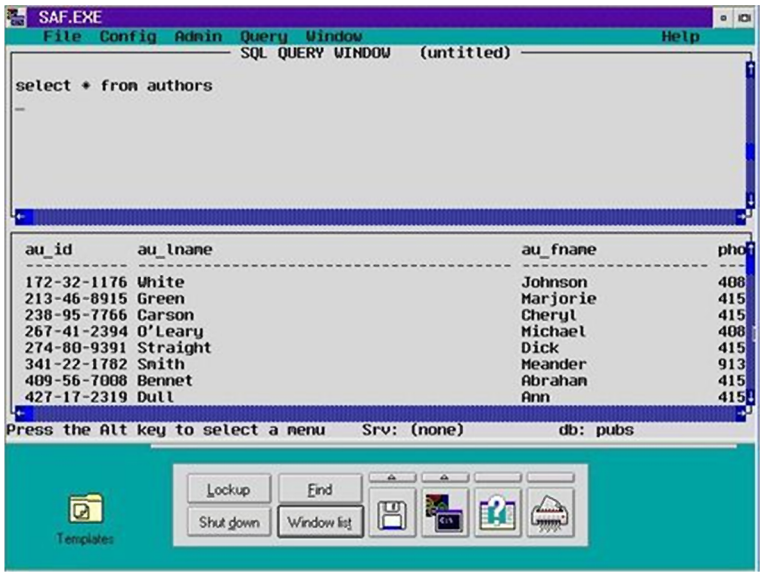
Su adopción por parte del mercado comenzó a partir de empresas pequeñas y medianas, en las que acostumbran (o acostumbraban) a abundar sistemas Windows, aunque ha ido ganando cuota de mercado en las últimas versiones a la par que mejoraba su rendimiento, seguridad y escalabilidad y se adoptaban soluciones Windows para servidores de producción también en grandes empresas. Esta última consecuencia, presumiblemente, se debe a la mejora de los sistemas servidores Windows en las últimas versiones. Destacar que en el momento de su entrada en mercado, MS-SQL Server no era un obstáculo a nivel económico tan importante como otros productos más consolidados en el mercado, lo que sin duda ayudó a su adopción inicial. En el 2001 era la base de datos más popular en sistemas Windows.

La versión de SQL implementada por SQL Server se conoce como T-SQL, o *Transact-SQL*, y es una variante del estándar SQL-92 adoptado por la empresa, aunque con pequeñas características propias tal y como introducen todos los vendedores de bases de datos.

5.2.1. Historia y evolución

Microsoft comenzó a desarrollar a finales de los ochenta junto con Sybase la primera versión de un motor de base de datos para competir con las grandes empresas hegemónicas en el mercado en aquella época, como Oracle o IBM. Fruto de este trabajo apareció SQL Server 1.0 para OS/2. Posteriormente apareció la versión 3.0 para Unix, pero no fue hasta 1992 cuando apareció la primera versión de Microsoft SQL Server como tal, la 4.2.

Figura 7. SQL Server 1.0 en OS/2



La siguiente versión que apareció fue la 6.0, para arquitecturas NT. A partir de esta versión, Microsoft negoció la exclusividad para el código de SQL Server en sus sistemas operativos, por lo que acaba la aventura conjunta con Sybase.

SQL Server 7.0 aparece en 1997, reescrita totalmente a partir del código original de Sybase y es la primera versión que incorpora una interfaz gráfica. Es un gran éxito comercial y se considera la primera versión propia totalmente de Microsoft.

A partir de este punto las versiones han sido SQL Server 2000, lanzado en el 2000, y el 2005, ambas en distintos paquetes con distinta funcionalidad para abarcar todo el espectro profesional del mercado. Estas son evoluciones "naturales" del producto según las demandas del mercado, logrando que en la actualidad sea un motor de base de datos fiable y con pocos problemas de seguridad conocidos. Las últimas versiones soportan arquitecturas de 64 bits.

5.2.2. Arquitectura

En contraste con otras bases de datos que corren en distintos sistemas operativos, SQL Server solo lo hace en Microsoft Windows, lo que determina muchas de sus características. Las posibilidades de combinación de sistema operativo–hardware es mucho menor que para algunos de sus competidores, co-

mo Oracle, en que esta combinación puede llegar a 26 casos. Es por esta razón por lo que en algunos aspectos este hecho puede repercutir directamente en la seguridad, como en algunos gusanos que utilicen desbordamiento de búfer para conseguir la ejecución de código malicioso y puedan aprovechar direcciones de memoria o llamadas a la API del sistema operativo sin necesidad de afrontar una casuística compleja.

En este apartado se explican las características principales de la arquitectura de SQL Server 2000. Tanto la versión anterior (7) como la posterior (2005) son muy similares y las diferencias se comentan cuando es necesario. Aunque existen diferencias, estas no son relevantes, ya que nuestro objetivo es presentar las principales características estructurales de las bases de datos en cuanto a seguridad se refiere.

El principal proceso de SQL Server es `sqlservr.exe` (servicio *MSSQLServer*), y por defecto escucha las peticiones en el puerto 1433. Se trata del servicio de la base de datos propiamente dicha, que accede y almacena los datos. La versión de SQL Server 2000 también escucha en el puerto 1434 UDP con el servicio SQL Server Monitor, el cual informa del estado del servidor a una petición concreta. Como veremos más adelante, este puerto proporciona posible información a un atacante, por lo que se desaconseja su acceso público. Estos puertos pueden cambiarse, lo que es una práctica recomendable en cualquier base de datos para evitar desvelar de forma inmediata el servicio que se encuentra tras un puerto. SQL Server dispone de una opción para "ocultarse" de escaneos, llamada *hide server*, mediante la que automáticamente pasa a escuchar del puerto 2433 y deja de responder a peticiones *broadcast*, aunque esta opción tiene algunos problemas cuando hay múltiples instancias del servidor.

Otro proceso importante es `sqlmangr.exe` (SQL Manager) cuya utilidad es arrancar y parar el servidor de la base de datos así como el agente SQL. El agente SQL (SQL Agent, servicio SQLServerAgent) es el proceso que se encarga de lanzar tareas y procedimientos de forma programada anteriormente, una especie de planificador para lanzar trabajos contra la base de datos. En la versión 2005, este proceso ha recibido importantes mejoras en aspectos de seguridad.

Finalmente, existen otros servicios en SQL Server:

- *Open data services*: Se tata de una interfaz entre las bibliotecas (*Net-libraries*) y las aplicaciones que utiliza el servidor. Su funcionamiento está intrínsecamente ligado con el de los procedimientos extendidos, que se explican posteriormente en este apartado.

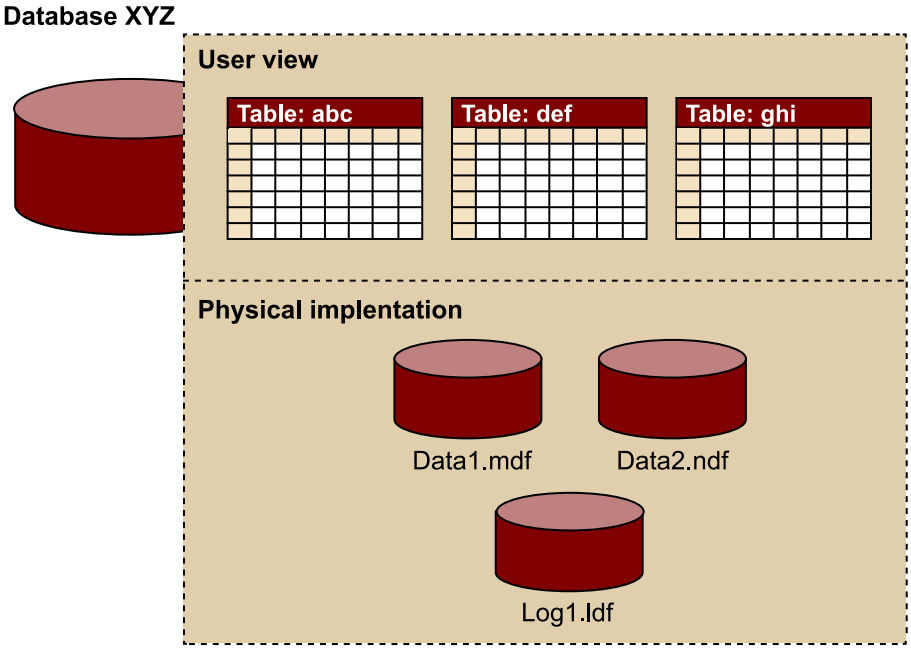
- *Microsoft search service*: Proporciona soporte para funcionalidades de búsqueda en texto y de indexación en tablas. Esta funcionalidad se implementaba como un servicio distinto en SQL Server versión 7, aunque posteriormente se integró con el servicio MSSQLServer.

- *Microsoft distributed transaction coordinator (MS DTC service)*: Servicio para coordinar transacciones entre distintos servidores implicados en una transacción distribuida.

Los datos físicos se guardan en archivos con las siguientes extensiones:

- mdf: Son los archivos primarios de la base de datos y apuntan al resto de archivos que forman parte de la misma.
- ndf: Son los archivos secundarios y guardan toda la información que no se guarda en los archivos primarios.
- lgf: Guardan toda la información de log relativa a la base de datos y utilizada para la recuperación de datos en un punto de tiempo fijado.

Figura 8. Tipos de ficheros



Cada instalación de SQL Server tiene cuatro bases de datos por defecto:

- *master*: Es la más importante y almacena la información de catálogo del resto de objetos de la base de datos.
- *model*: Guarda las plantillas para la creación de todas las bases de datos del sistema.
- *tempdb*: Guarda información temporal de tablas y de procedimientos.
- *msdb*: La utiliza el servidor para lanzar alertas y trabajos programados con el SQL server agent.

En cuanto a las tablas más importantes del catálogo de SQL Server, el esquema se almacena en la base de datos *master*. Algunas de las tablas más destacables son:

- sysobjects: Almacena todas las tablas de las bases de datos.
- syscolumns: Almacena las columnas de las tablas de las bases de datos.

- sysusers: Información acerca de los usuarios de la base de datos.
- sysdatabases: Bases de datos disponibles.
- sysxlogins: Información acerca de las cuentas de usuario.

SQL Server dispone de varios roles de usuario predefinidos que tienen unos permisos preestablecidos. Estos se dividen en **roles de servidor**, que contemplan aspectos de la administración del servidor de base de datos, y **roles de base de datos**, que contemplan aspectos referentes a las bases de datos en sí. Dichos roles no se pueden cambiar, crear ni borrar. Los más destacables de servidor son:

- dbcreator: Permite la creación y administración de bases de datos.
- diskadmin: Acceso y administración de dispositivos de almacenamiento físicos.
- securityadmin: Permite la creación y administración de roles y usuarios.
- sysadmin: Control administrativo total sobre el servidor.

En cuanto a los roles de base de datos, los más destacables son:

- db_datareader: Permite el acceso de lectura a una base de datos.
- db_datawriter: Permite la modificación, creación y borrado de datos.
- db_owner: Permite realizar cualquier operación en la base de datos.
- db_securityadmin: Permite la administración de roles y de permisos de objetos.

SQL Server permite la creación de roles de usuario a los que se permite asignar permisos de modo personalizado con cualquier objeto de la base de datos.

La autenticación en SQL Server se realiza mediante dos mecanismos: autenticación nativa de la base de datos y autenticación mediante el sistema operativo. También se proporcionan tres fórmulas para este proceso:

- nativa,
- sistema operativo,
- mixta.

El hecho de poder utilizar la **autenticación del sistema operativo** es alguna de las características derivadas de su relación con el mismo, así como de no tener que ser flexible como para poder ejecutarse en otros sistemas operativos que no sean de Microsoft.

En el caso de la **autenticación nativa** de SQL Server, los datos que se transmiten al servidor no se cifran, únicamente se ofuscan mediante una transformación reversible, por lo que, en caso de no realizarse la autenticación de forma

local o mediante una conexión segura, pueden ser interceptados y capturados sin dificultad. Es recomendable utilizar la autenticación que proporciona mediante el sistema operativo o asegurar que la comunicación siempre es a través de un canal cifrado.

Versión 2000

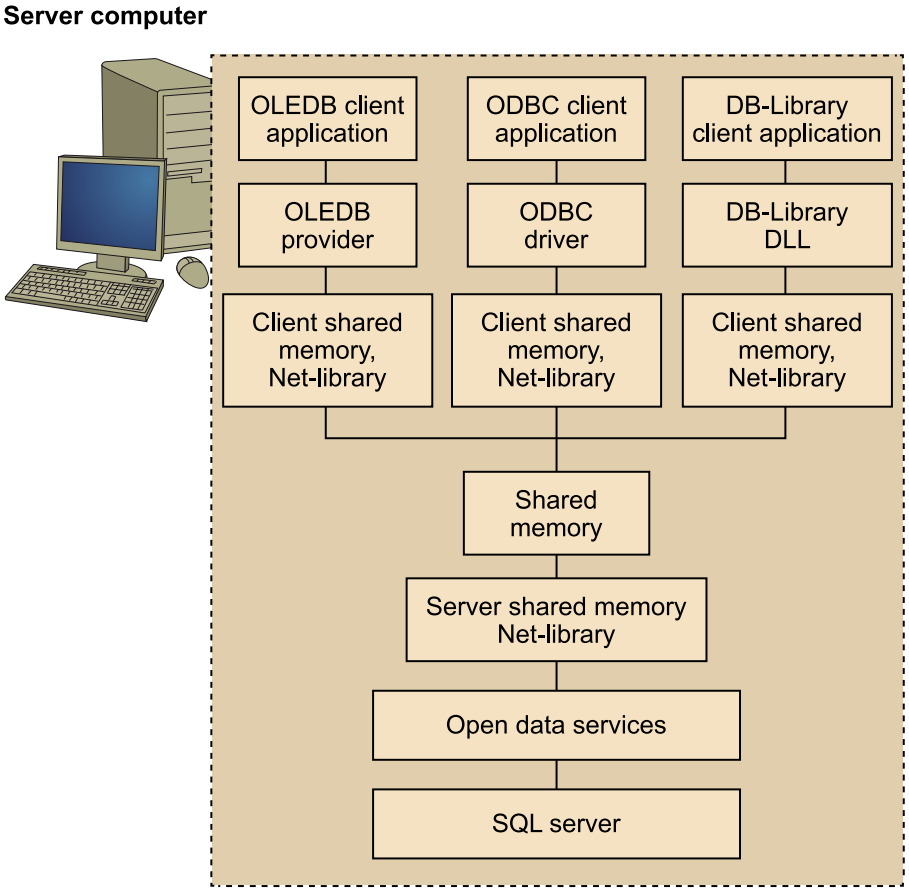
Este problema se da en la versión 2000 de la base de datos, ya que la 2005 proporciona nuevas funcionalidades para solucionar este problema, incluso proporcionando la opción de crear certificados digitales para la autenticación de usuarios. Además, proporciona funcionalidad para exigir una mínima complejidad y longitud en las contraseñas, así como un tiempo de expiración para su renovación regular.

La **autenticación mixta** simplemente mezcla ambos sistemas y realiza el proceso en dos fases: primero comprueba las credenciales mediante el sistema operativo y posteriormente con las credenciales propias para el usuario de base de datos.

TDS (*tabular data stream protocol*) es el protocolo que utiliza SQL Server para la comunicación cliente–servidor, incluyendo la autenticación y las peticiones y respuestas de consultas a la base de datos. Existen implementaciones libres de este protocolo más allá de la proporcionada por Microsoft.

Network libraries son las bibliotecas que utiliza SQL Server para la comunicación por red con los clientes. Por defecto la comunicación se realiza mediante TPC/IP, aunque también es posible especificar comunicación mediante *named-pipes* (en este caso más lenta y únicamente en el mismo servidor), mediante *shared memory* (memoria compartida) en la que la comunicación también es local pero obtiene la máxima velocidad disponible o mediante SSL (*secure socket layer*) para cifrar la comunicación entre ambos extremos. Existen varias bibliotecas adicionales que soportan protocolos menos comunes, siendo la mayoría de ellos antiguos y únicamente existentes por cuestiones de compatibilidad.

Figura 9. Protocolos soportados para comunicación cliente-servidor



Los procedimientos almacenados en SQL Server tienen la misma funcionalidad que en otras bases de datos, que es la de extender la capacidad de las consultas mediante la creación de procedimientos mediante el lenguaje de programación TSQL. Además, SQL Server proporciona los procedimientos almacenados extendidos (*XSP eXtended Stored Procedures*) que permiten crear nueva funcionalidad mediante la creación de procedimientos en lenguaje de alto nivel con C o C++ y almacenarlos en *dlls* para que pueda acceder a dichas funciones la base de datos mediante la API de *open dataservices* .

Algunos de los procedimientos almacenados de SQL Server son el primer objetivo de un atacante a la base de datos, debido a la gran potencia de los mismos, como veremos más adelante. Cabe destacar que procedimientos como *xp_cmdshell* permiten la ejecución de comandos directamente contra el sistema operativo de la máquina que alberga la base de datos, por lo que se entiende rápidamente por qué estos procedimientos son el primer objetivo de un ataque. Por ello, es recomendable no dar permisos a ningún usuario que no sea el administrador, o eliminarlos completamente en caso de no ser necesarios. En caso de su eliminación de la base de datos, es buena idea el borrado de las *dll* que proporcionan la funcionalidad para evitar que vuelvan a incluirse dentro de las funcionalidades de la base de datos.

SQL Server proporciona un mecanismo de encriptado de los procedimientos almacenados para evitar la consulta del código de los mismos.

5.3. MySQL

5.3.1. Introducción

MySQL es una base de datos muy popular hoy en día a pesar de ser una de las últimas en aparecer en el mercado a pesar de la falta de muchas funcionalidades hasta versiones muy recientes. Según la compañía, hay más de 10 millones de instalaciones de su producto en la actualidad. De hecho y desde un punto de vista purista, MySQL ha carecido de algunas de las principales características consideradas necesarias para un motor de bases relacional. Sin embargo, su popularidad se debe a la conjunción del boom de Internet y a ser una base de datos de código libre, aunque este aspecto se matiza más adelante.

La falta de características funcionales de integridad no ha sido un obstáculo para su popularidad. En un momento de aparición de gran cantidad de aplicativos bajo licencias libres de todo tipo, la posibilidad de disponer de una base de datos bajo estas mismas condiciones y con un rendimiento aceptable para las operaciones menos complejas fue todo un hito.

A partir de ese punto, MySQL fue cobrando popularidad con gran rapidez, pasando a un nuevo modelo de negocio en las versiones más recientes. Hasta entonces sus ingresos se basaban en las certificaciones y el cobro de licencias para empresas que deseaban usar el producto bajo licencia no GPL. Sin embargo, a partir de la versión 5 se creó un nuevo concepto con la versión de MySQL Enterprise, que incluye además del motor de base de datos relacional un soporte continuo para clientes, ya pensado para un mercado más empresarial en el que este tipo de servicios son necesarios. Para no perder sus raíces, MySQL continua en paralelo el desarrollo de MySQL 5 Community Edition en un modelo parecido al adoptado por Red Hat con Fedora, siendo la diferencia entre ambas versiones la disponibilidad de binarios compilados y optimizados para distintas plataformas, que dejan de estar disponibles para la versión Community, así como el compromiso de la publicación de versiones nuevas puntualmente cada mes con todos los *bugs* corregidos para la versión Enterprise. El código continúa disponible para su descarga bajo licencia GPL.

A pesar de su falta de implementación de características básicas en cualquier base de datos relacional, como la integridad referencial, MySQL ha ido evolucionando poco a poco hasta llegar a un punto en que dispone de las mismas funcionalidades básicas que cualquier otra base de datos comercial. Sin embargo, es importante no olvidar que el camino de su popularización fue su licencia libre, su capacidad para correr en prácticamente cualquier plataforma y su funcionamiento correcto como base de datos ligera para las funcionalidades más básicas. En particular, lo que se conoce como plataforma LAMP

(con sus variaciones) que ha sido uno de los pilares de creación de gestores de contenido en Internet (LAMP: Linux + Apache + MySQL + PHP), permitiendo prácticamente a cualquiera la creación de aplicativos con una inversión mínima. MySQL fue el complemento ideal a una serie de herramientas de similar filosofía y que irrumpió en el mercado en el momento adecuado. Hoy en día se encuentra en multitud de gestores de contenido libres, por lo que su conocimiento en cuanto a seguridad es muy importante por su gran presencia en el mercado. Es más, debido a su historia y el entorno de popularización, suele estar presente en muchas ocasiones en la DMZ de las compañías, constituyendo un riesgo potencial en caso de no estar debidamente securizado.

5.3.2. Historia y evolución

MySQL es el nombre de la base de datos perteneciente a MySQL AB, que es la compañía sueca que la creó (AB es el equivalente a S. A. en castellano). Desde el momento de su creación, la compañía ha querido instaurar una filosofía de empresa parecida al famoso "*don't be evil*" de Google, destacando una serie de valores que se vieran reflejados en su comportamiento y productos y que se resume en una única frase enunciada por ellos mismos: "*To make superior data management* software *available and affordable to all*" (Crear software de administración de datos de calidad superior, disponible para todo el mundo). La compañía se fundó por David Axmark, Allan Larsson y Monty Widenius. El nombre de la base de datos es por el hijo de Monty, llamado My (nombre sueco).

Todas las versiones de MySQL están disponibles para gran cantidad de plataformas, lo que no es sorprendente siendo un producto de código libre. Siempre es posible intentar la compilación para nuevas plataformas, pero en la actualidad está disponible en su versión Enterprise para Red Hat Enterprise Linux, Novell SuSE Enterprise Linux, Debian GNU/Linux, Sun Solaris 10 y 9 y 8, HP-UX 11.23, Windows NT/2000, Windows XP, Windows 2003 Server, Apple Mac OS X v10.4, IBM AIX, OpenServer 6.0,Fedora, openSUSE, CentOS y Ubuntu. La versión Community está disponible todavía para más plataformas, por lo que queda patente la flexibilidad del motor de base de datos.

El código inicial fue creado por Monty a finales de los ochenta, principios de los noventa para uso propio. Según fue creciendo el proyecto y añadiendo nuevas funcionalidades, empezó a pensar en fundar una pequeña compañía para la comercialización del producto. A mediados de los noventa funda MySQL AB y se buscan inversores de capital riesgo para invertir en el desarrollo del producto, dando paso a las sucesivas versiones cada vez con mayor funcionalidad, si bien hasta la versión 5 no se han implementado características consideradas como muy importantes, como pueden ser los procedimientos almacenados, *triggers* o cumplimiento de las propiedades ACID (atomicidad, consistencia, isolación y durabilidad) consideradas básicas para bases de datos relacionales. Es en esta época cuando la compañía se traslada a California.

La adquisición de la compañía por parte de Oracle puso toda esta filosofía en jaque. Desde entonces existe una *community edition* de código abierto y otra versión únicamente de pago y que implementa las últimas características. Esto ha hecho que haya perdido parte de su público que han decidido migrar a otras soluciones de código abierto. Uno de los fundadores de MySql decidió abandonarla para crear una nueva base de datos que siguiese la misma filosofía inicial de MySql.

5.3.3. Arquitectura

El principal programa de la base de datos es *mysqld*, aunque dependiendo del entorno puede ser *mysqld-nt*, que es el equivalente para entornos Windows, o *mysqld-max*, en caso de que incorpore la base de datos MaxDB. Se trata de un soporte a otro tipo de base de datos que también desarrolla MySQL AB, con un impacto en el mercado muy marginal y que no trataremos. Este programa es el que se encarga del acceso y tratamiento del motor de la base de datos, el resto son programas cliente.

Hay distintos métodos para la comunicación cliente-servidor:

- TCP/IP, tanto para conexiones locales como remotas.
- *Named pipes*, en conexiones remotas solo en sistemas Windows.
- *Sockets*, en conexiones remotas sólo en sistemas Unix.

En conexiones TPC/IP, el servidor escucha por defecto en el puerto 3306. MySQL no utiliza ningún protocolo "extraño" de red, y a partir de la versión 4.0 soporta el uso de SSL en conexiones de red.

Existen distintas interfaces para interactuar con la base de datos como la librería C llamada *libmysqlclient*, el conector ODBC o el conector JDBC.

APIs de MySQL

Aparte de estas interfaces, que son las que soporta oficialmente MySQL, existen multitud de APIs desarrolladas de forma independiente con soporte para PHP, Perl, Python, Ruby, Pascal y Tcl.

En cuanto al tratamiento de los archivos por parte de MySQL, se asocia cada base de datos con un subdirectorio bajo el directorio de almacenamiento de datos, especificado en el archivo de configuración de MySQL (normalmente en el archivo *my.cnf*). Cada subdirectorio tiene el nombre de la base de datos que representa. Una base de datos puede estar vacía y puede no contener ningún subdirectorio. Cada tabla dentro de la base de datos se representa en un fichero con extensión .frm que contiene la definición de la tabla. En cuanto al almacenamiento de los datos propiamente dichos, depende del tipo de tablas, y aquí es cuando entra en juego una de las características más llamativas de MySQL.

A la hora de crear una tabla, es posible especificar el tipo de la misma, de modo que el motor de base de datos correspondiente será el que se encargue de tratarla. Esto implica que en función de dicho tipo, las tablas tendrán distintas características, lo que puede resultar confuso. La mejor aproximación es pensar como si existiesen distintas bases de datos en una, y a la hora de definir las

tablas el usuario elije con qué motor quiere trabajar. En función de dicho tipo, también se guardarán los ficheros en disco con un formato u otro. MySQL soporta los siguientes tipos de tabla:

- MyISAM: Se trata del tipo de tabla más popular y el tipo de tabla "original" de MySQL disponible desde las primeras versiones. El motor que se encarga de tratar este tipo carece de muchas características deseables para bases de datos relacionales, por lo que no dispone de funcionalidad como integridad referencial. Estas tablas se representan en disco como archivos con extensión .MYD para almacenar los datos y archivos con extensión .MYI para almacenar los índices.

- InnoDB: Este motor es el que, en buena parte, ha permitido que MySQL haya evolucionado hasta ser una base de datos "seria" implementando características de las que carecen los otros motores. Es compatible con las propiedades ACID y permite transacciones atómicas, por lo que se pueden confirmar o refutar con Commit y Rollback respectivamente. También implementa integridad referencial. En este caso, las tablas no se almacenan en disco usando distintos archivos, sino que se guardan todos los datos en un único espacio de almacenamiento que utiliza el motor de InnoDB y que puede consistir en uno o varios archivos, según va creciendo de tamaño. Por defecto, el nombre del espacio en el que InnoDB almacena los datos es *ibdata1*, y donde almacena los logs *ib_logfile0* y *ib_logfile1*.

Tanto InnoDB como MyISAM son los principales motores de MySQL, los que se listan a continuación tienen una utilidad más marginal:

- Merge: Las tablas de este motor son una colección de tablas MyISAM idénticas, representadas en disco con la extensión .frm para el formato de la tabla y .MRG para listar todas las tablas MyISAM que forman parte de la tabla Merge. En esencia, se trata de crear una entidad que aglutina un conjunto de tablas MyISAM idénticas con la finalidad de sobrepasar las limitaciones de espacio de las primeras.

- BDB (Berkeley DB): Las tablas en disco tienen una extensión .frm para el formato y .db para almacenamiento de datos e índices. Este motor implementa las propiedades ACID y soporta transacciones atómicas, así como bloqueo a nivel de página, lo que puede provocar problemas de bloqueo como precio para un aumento de la concurrencia.

- Memory tables: Se trata de tablas que, aunque tienen la definición en disco en un archivo con extensión .frm, guardan los datos e índices en memoria, resultando en el tipo de tablas más rápido disponible. Al usar un recurso escaso, no es un tipo factible para tablas grandes. Utiliza bloqueo a nivel de tabla, lo que baja la concurrencia y evita bloqueos.

En la versión 5 de MySQL se introducen nuevos motores adicionales a estos. Cabe destacar que es posible crear un motor propio y utilizarlo en MySQL mediante un proceso relativamente sencillo, lo que favorece la aparición de nuevos motores con el paso del tiempo, especializados en distintas funcionalidades. Los nuevos motores disponibles son el Example, Federated, Archive, CSV y Blackhole.

Como se ha visto, el motor de MySQL permite trabajar con distintos tipos de bases de datos con características propias. Sin embargo, cabe destacar que recientemente dos de las compañías que trabajaban conjuntamente con MySQL en el desarrollo de estas bases de datos han sido adquiridas por la competencia. En concreto, Oracle compró en 2005 Innobase OY, la compañía finlandesa encargada del desarrollo del motor InnoDB. Teniendo en cuenta que este motor era el que implementaba un mayor número de funcionalidades necesarias para una base de datos relacional, implementando la integridad referencial desde las primeras versiones, puede suponer un revés importante para MySQL, aunque han llegado a un acuerdo para continuar el desarrollo conjunto durante un tiempo.

Un caso similar ha ocurrido con la adquisición, también por Oracle, de Sleepycat Software, la compañía creadora de la Berkeley DB y también usada por MySQL.

En cuanto al esquema de las bases de datos, desde la versión 5 MySQL incorpora la base de datos Information Schema, que incorpora las tablas en las que se almacena la información respecto al resto de bases de datos y tablas (metadatos). Algunas de las tablas más interesantes son:

- Tables: Información acerca de las tablas de otras bases de datos.
- Columns: Campos que componen las tablas.
- Views: Vistas disponibles.
- User_privileges: Permisos de los que disponen los usuarios.
- Routines: Almacena procedimientos y funciones.

En versiones anteriores a la 5 esta base de datos no existía, por lo que se debían consultar estos datos en la base de datos llamada *mysql*. Aunque no se recogía información en la misma respecto a las tablas que conformaban otras bases de datos, se guardaba información respecto a los usuarios, incluyendo los *hashes* de las contraseñas. En concreto, esta información estaba disponible en la tabla *user*.

Otras tablas

Otras tablas interesantes eran *db*, *tables_priv* y *host*.

MySQL utiliza un sistema propio para la autenticación, consistente en el uso de un par usuario-contraseña. Permite especificar junto con el nombre de usuario el *host* desde el que se permite su autenticación, por lo que un intento de conexión desde otro dispositivo no permitiría entrar en el sistema a pesar de conocer tanto el usuario como la contraseña correcta. No permite ninguna interacción con el sistema operativo para la autenticación.

A nivel histórico, han existido problemas con la autenticación en MySQL. En concreto, en versiones anteriores a la 4.1 el conocimiento del *hash* de la contraseña era suficiente para lograr la autenticación, sin necesidad de ningún "crackeo". En versiones anteriores a la 3.23.11, existía una debilidad que permitía lograr la autenticación con un único carácter, por lo que el historial de la base de datos es malo en cuanto a problemas de autenticación, lo que se suma a que en versiones antiguas no se utilizaba ningún tipo de cifrado en la autenticación por red, por lo que los datos podían ser interceptados por cualquiera.

Las funciones definidas por el usuario (UDF, *user definded functions*) son extensiones que permiten la creación de nuevas funcionalidades mediante la programación en lenguajes de alto nivel (C, C++) y su importación desde la base de datos. Aunque este punto se trata con más detalle en próximos capítulos, cabe decir que la potencia que aporta es enorme, así como los riesgos en caso de no ser cuidadosos.

El tratamiento en el almacenamiento de datos que se utiliza en MySQL es bastante particular, no es frecuente encontrar la correspondencia tan directa entre bases de datos y tablas con archivos en disco. Esto implica que es necesario un cuidado especial a la hora de securizar los permisos sobre estos archivos, porque el acceso a los mismos por parte de un usuario no autorizado implica la disponibilidad directa de los datos, siendo factible incluso la modificación de los mismos sin demasiada dificultad.

En cuanto a los permisos en la base de datos, no existen roles predefinidos, sino que los permisos se asignan directamente a los usuarios. Hay que destacar que una vez que se cambian los privilegios, es necesario especificar de modo explícito que se apliquen mediante la orden `FLUSH PRIVILEGES`, ya que hasta entonces no se producen los cambios. Los permisos pueden ser a dos niveles:

- El primer nivel es en cuanto a los **objetos de la base de datos**, siendo esto lo habitual en otras bases de datos.

- El segundo nivel es en cuanto a **permisos** de ejecutar ciertas acciones **con el usuario** en cuestión, como acceder a disco para cargar datos de un archivo concreto o para guardar los resultados de una consulta en otro. En siguientes módulos se explica cómo esta característica es una de las más llamativas y que se debe tener muy en cuenta a la hora de securizar la base de datos correctamente.

5.4. DB2

DB2 Universal Database, creado por IBM, se considera por algunos como la primera base de datos en utilizar SQL. Se trata de una de las bases de datos más veteranas en la industria.

Aunque esta base de datos tiene un porcentaje de mercado importante, da la impresión de tener una presencia menor a la que muestran las estadísticas, tal vez porque no está habitualmente expuesta a entornos más "externos" como Internet sino a entornos más fortificados e internos de la estructura de red de la compañía. A pesar de su presencia, es difícil encontrarla habitualmente como ocurre con otras bases de datos. Posiblemente el escenario más habitual en el que se encuentra esta base de datos es en conjunción con otros productos de IBM, sobre todo en los famosos servidores AS/400 que adquirieron gran popularidad a mediados de los noventa en el entorno empresarial y en el que la base de datos de IBM estaba instalada por defecto.

IBM se distingue del resto del mercado al proporcionar algunas características (al menos a nivel de marketing) avanzadas que quedan lejos de otros motores de bases de datos, como *data-warehousing* o *data-mining*. Es posible que el motor de DB2 para z/OS en arquitecturas de 64 bits sea al más potente para este tipo de operaciones del mercado, dando un rendimiento muy bueno y reconocido tanto por el mercado como por la competencia. Es, por tanto, un motor de base de datos algo "exclusivo" y con unas características bastante concretas que tal vez lo alejan de un tipo de cliente más generalista. Hay que considerar que las soluciones de IBM normalmente combinan tanto software como hardware, constituyendo una solución integral debida al modelo de negocio de la empresa y que lo pueden alejar de otro tipo de motores de bases de datos. Aun así, IBM ha proporcionado versiones para plataformas generalistas en sus últimas versiones, mostrando un cambio en su estrategia empresarial.

En cuanto a seguridad se refiere, DB2 tiene los mismos problemas que cualquier otra base de datos. Sin embargo, siempre ha tenido una buena reputación debido por una parte a que proporciona una funcionalidad más reducida en comparación con otras bases de datos, lo que supone una menor superficie de ataque, y minimiza las posibilidades de sufrirlo. Por otra parte, IBM ha tenido una política muy eficiente en cuanto a solución de problemas reportados, proporcionando soluciones de manera rápida y eficaz. Todo esto, sumado al poco impacto conocido, al menos en cuanto a problemas de seguridad graves se refiere, ha resultado en una buena imagen de robustez de este motor de base de datos.

5.4.1. Historia y evolución

Aunque fue en 1978 cuando IBM lanzó el famoso System R, no fue hasta el año 1982 cuando IBM lanzó junto a su plataforma *mainframe* los productos SQL/DS y DB2. Aunque DB2 es una evolución del System R, hasta esta fecha no se creó como producto propio con el nombre que ha llegado hasta la actualidad.

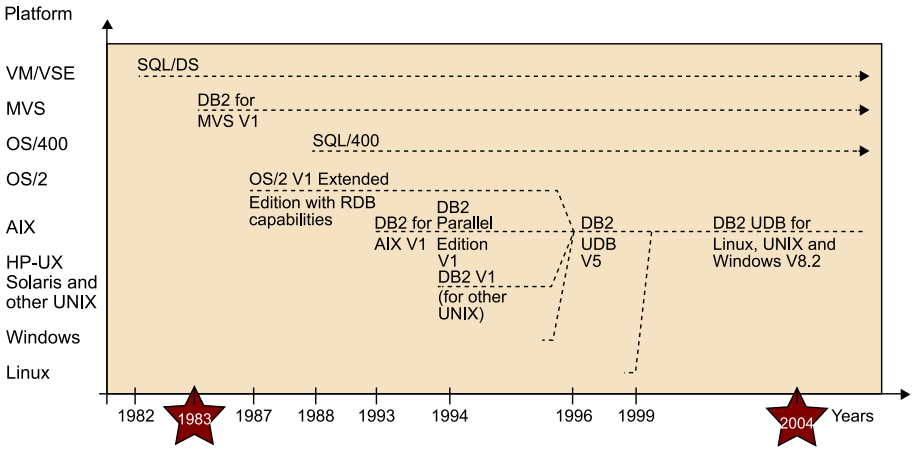
Inicialmente, y siguiendo la filosofía de IBM, la base de datos estaba únicamente disponible para los *mainframes* de su misma marca. Durante la década de los noventa se produjo un cambio en la estrategia de la compañía, ampliando la disponibilidad de la base de datos para otras plataformas, concretamente

para servidores Windows y Unix, aunque posteriormente se lanzó para plataformas Linux. Durante este recorrido la alternancia en la cabeza de cuota de mercado ha sido entre DB2 y Oracle de modo histórico.

DB2 está disponible en tres ediciones con distintas funcionalidades, conocidas como Express, Workgroup y Enterprise (de menos a más). También permite licenciar una versión reducida en la que se puede abaratar el coste mediante la eliminación de servicios no requeridos por el cliente, así como una versión avanzada conocida como Data Warehouse Enterprise Edition. Existe una versión gratuita conocida como DB2 9 Express C, que apareció en el año 2006 como respuesta a los productos que ofrecía la competencia.

Las versiones que se encuentran en el mercado en la actualidad oscilan desde la versión 7 hasta la 9, aparecida en el mercado durante el 2006. La principal característica anunciada por IBM de su última versión de DB2 es el almacenamiento nativo de XML.

Figura 10.Timeline de DB2



Se puede dividir el producto en tres ramas principales en función de las plataformas en las que corren, cada una de las cuales con un núcleo común aunque con algunas peculiaridades y características propias:

- DB2 para z/OS.
- DB2 para Windows, Unix y Linux (o también conocido como LUW).
- DB2 para iSeries (AS/400).

En el 2001 Oracle adquirió Informix, incorporando una variada funcionalidad de este motor de base de datos en DB2, sobre todo la relacionada con el concepto de base de datos orientada a objetos.

5.4.2. Arquitectura

Procesos

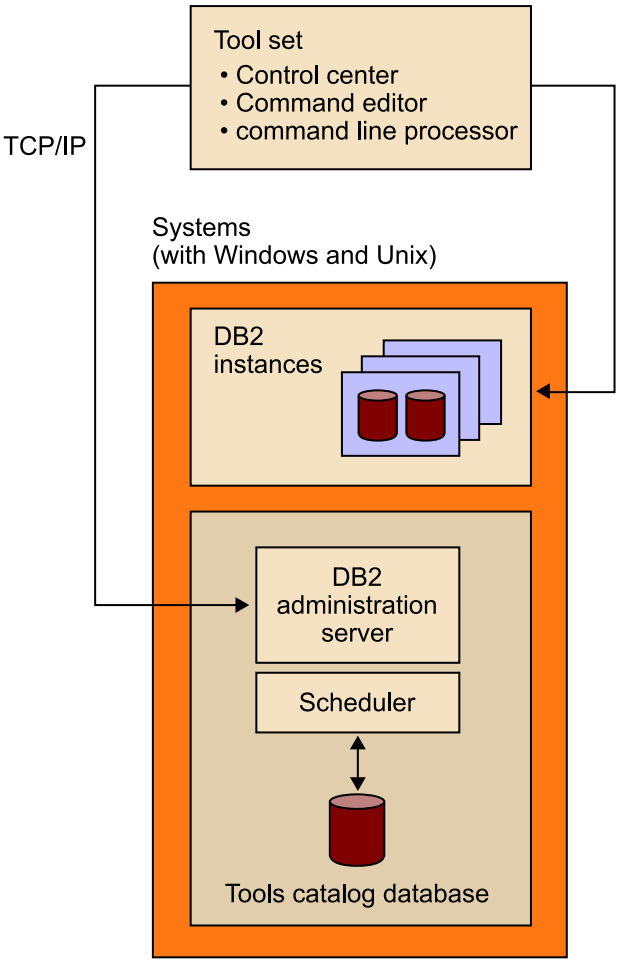
En cada uno de los equipos en los que se instala un servidor de base de datos DB2 pueden correr varias instancias del mismo, así como en cada instancia puede haber distintas bases de datos. En una instalación por defecto, corren en el servidor dos procesos que se corresponden con dos instancias del servidor, llamados DB2 y DB2CTLSV. En caso de instalar la base de datos de ejemplo disponible en la instalación del producto, esta se encuentra en la instancia llamada DB2.

Aparte de las instancias, existe el servicio DB2, *database administration server* o DAS, que se encarga de la administración de la base de datos.

El nombre con los que los procesos corren en Windows son DB2SYSCS para las instancias y las bases de datos. El nombre de este proceso en Linux es DB2SYSC. El proceso de administración de la base de datos o DAS se llama DB2DASRRM en ambos entornos.

Existe un tercer proceso llamado DB2FMP, en el que se cargan las rutinas definidas en la base de datos. Se cargan en un proceso separado para que, en caso de producirse algún problema o en caso de ejecutarse alguna rutina defectuosa, no caiga el proceso principal de la base de datos.

Figura 11. Relación del DAS con el resto de componentes



Cada una de las instancias escucha de puertos distintos, por defecto DB2 escucha del puerto 50000 y DB2CTLSV en el puerto 50001. En entornos Windows es posible utilizar *namedpipes* para la comunicación. Estos puertos son configurables.

En cuanto al DAS, el puerto por el que escucha por defecto es el 523 (tanto TCP como UDP).

Para la comunicación entre el servidor y el cliente, DB2 usa DRDA (*distributed relational database architecture*), el cual es un protocolo abierto aunque no ha gozado de gran difusión. Dicho protocolo se transmite sobre TCP/IP y dentro de cada uno de dichos paquetes se encuentran empaquetados uno o más paquetes DSS (*datastream structures*), en los que se encuentran los comandos de base de datos junto con sus parámetros correspondientes.

En la autenticación, el protocolo realiza la transmisión de credenciales en texto plano (aunque puede cambiar en función de la configuración del servidor), aunque en caso de esnifar los paquetes de red esto no parece evidente debido a que no se transmiten en ASCII sino que DB2 utiliza caracteres codificados mediante EBCDIC (*extended binary coded decimal interchange code*), código pro-

piedad de IBM. Dado este escenario, es posible realizar un mapeo entre ambos conjuntos de caracteres y acceder a los datos de autenticación en caso de acceder al tráfico entre cliente y servidor durante el proceso de autenticación.

Ficheros

La instalación de la base de datos por defecto se realiza en el directorio c:/Archivos de programa/IBM/SQLLIB en Windows, en el cual se pueden encontrar los ejecutables así como algún archivo de log. A partir de este directorio, se crea un subdirectorio para cada una de las instancias que se instalan del servidor. En Linux se instala por defecto en /opt/IBM/db2.

Las bases de datos se instalan en la raíz de la unidad en el caso de Windows, mediante la creación de un directorio para cada una de las bases de datos (por ejemplo, c:/DB2). A partir de cada uno de estos subdirectorios, se crea otro llamado NODE0000 y bajo el mismo, una serie de subdirectorios con un rango de nombres desde SQL00001 hasta SQL0000X, así como otro llamado SQLDBDIR. En esta estructura DB2 crea la serie de archivos utilizados por la base de datos y en los que contiene la información propiamente contenida en las mismas. El usuario con el que corre en Windows la base de datos es db2admin.

En cuanto a Linux, la estructuración de la información está íntimamente ligada con la creación de una serie de cuentas. Mediante la instalación por defecto se crean tres cuentas:

- *dasusr1*:Es la encargada de ejecutar el servicio de administración y es la que contiene los archivos ejecutables relacionados con el mismo así como los ficheros de log.

- *db2fenc1*: Es la encargada de ejecutar el servicio DB2FMP para las rutinas y funciones utilizadas por la base de datos.

- *db2inst1*: Es la encargada de ejecutar el servicio de la instancia de la base de datos, y en su directorio /home se encuentra el directorio /db2inst1, que a su vez contiene los directorios sqllib en el que se encuentran archivos de configuración de la instancia, y db2inst1, en el que se encuentran los archivos que contienen los archivos con los datos de la base de datos.

Estructura lógica

Los objetos de la base de datos se guardan en esquemas (*schemas*), lo que incluye tablas, vistas, procedimientos, etc. Los principales esquemas son:

- SYSIBM: información respecto a las tablas.
- SYSCAT: información respecto a las vistas.
- SYSFUN: información respecto a las funciones.

- SYSPROC: información respecto a los procedimientos.

Autenticación

La autenticación en DB2 se realiza únicamente mediante el sistema operativo. Esto proporciona ciertas ventajas comparado con otras bases de datos, como blindaje respecto a configuraciones incorrectas, como las relacionadas con las instalaciones por defecto de MS SQL Server del usuario *sa* sin ninguna contraseña, o las relacionadas con la gran cantidad de cuentas heredadas en Oracle con sus correspondientes contraseñas y que en muchas instalaciones no se cambian por parte de los administradores. Aun así, en versiones antiguas de DB2 las cuentas que se creaban relacionadas con el motor de la base de datos usaban contraseñas por defecto, lo que comportaba el mismo problema. Este comportamiento implica que no existe información relacionada con los usuarios en DB2 en la que se almacenen las contraseñas de autenticación.

Es posible configurar el nivel de cifrado entre el cliente y el servidor a la hora de trasmitir los datos de autenticación, ya que como se ha visto anteriormente los datos pueden enviarse sin autenticar en caso de una configuración incorrecta.

Una vez los usuarios están autenticados, sus permisos se especifican mediante una serie de roles, igual que en otros motores de bases de datos, aunque en DB2 se conocen como "autoridades" (*authorities*). Las autoridades pueden tener un alcance de sus permisos respecto a la instancia o de la base de datos. Esta información se guarda en tablas en la base de datos dentro del esquema SYSCAT:

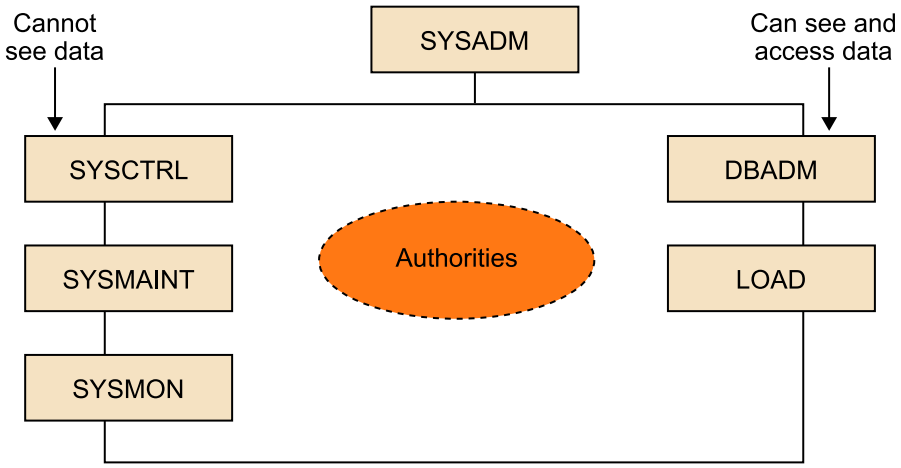
- DBAUTH: Información respecto a las autoridades de bases de datos y las acciones permitidas.

- TABAUTH: Información respecto a permisos sobre acciones sobre tablas de la base de datos.

- ROUTINEAUTH: Información respecto a quién tiene permisos de ejecución de las rutinas.

En DB2 el rol con mayores permisos es SYSADM, y el grupo al que pertenece es el SYSADM_GROUP.

Figura 12. Esquema principales autoridades



Tablas destacables

Las tablas más interesantes que contienen información respecto a los objetos de la base de datos son las siguientes:

- SysIBM.tables: Información respecto a las tablas de la base de datos.
- SysIBM.views: Información respecto a las vistas de la base de datos.
- SysIBM.columns: Información respecto a las columnas de las tablas de la base de datos.
- SysIBM.usernames: Información respecto a conexiones externas de la base de datos.

En DB2 la tabla comodín es *SYSIBM.Sysdummy1* (equivalente a la tabla *dual* en Oracle).

Bibliografía

http://attrition.org/dataloss/

http://math.hws.edu/vaughn/cpsc/343/2003/history.html

http://en.wikipedia.org/wiki/SQL_slammer_(computer_worm)

http://www.oracle.com/technology/deploy/security/alerts.htm