Ataques a Aplicaciones de Bases de Datos

Esteban Martínez Fayó

Argeniss (www.argeniss.com)

ekoparty security conference

Noviembre 2007

Buenos Aires, Argentina

Agenda

- Introducción a la seguridad en Bases de datos
- Importancia de la seguridad de aplicaciones web
- Vulnerabilidades de SQL Injection en aplicaciones web
- Herramienta de Demostración: SQLIT
- ¿Cómo protegerse?
- Conclusiones

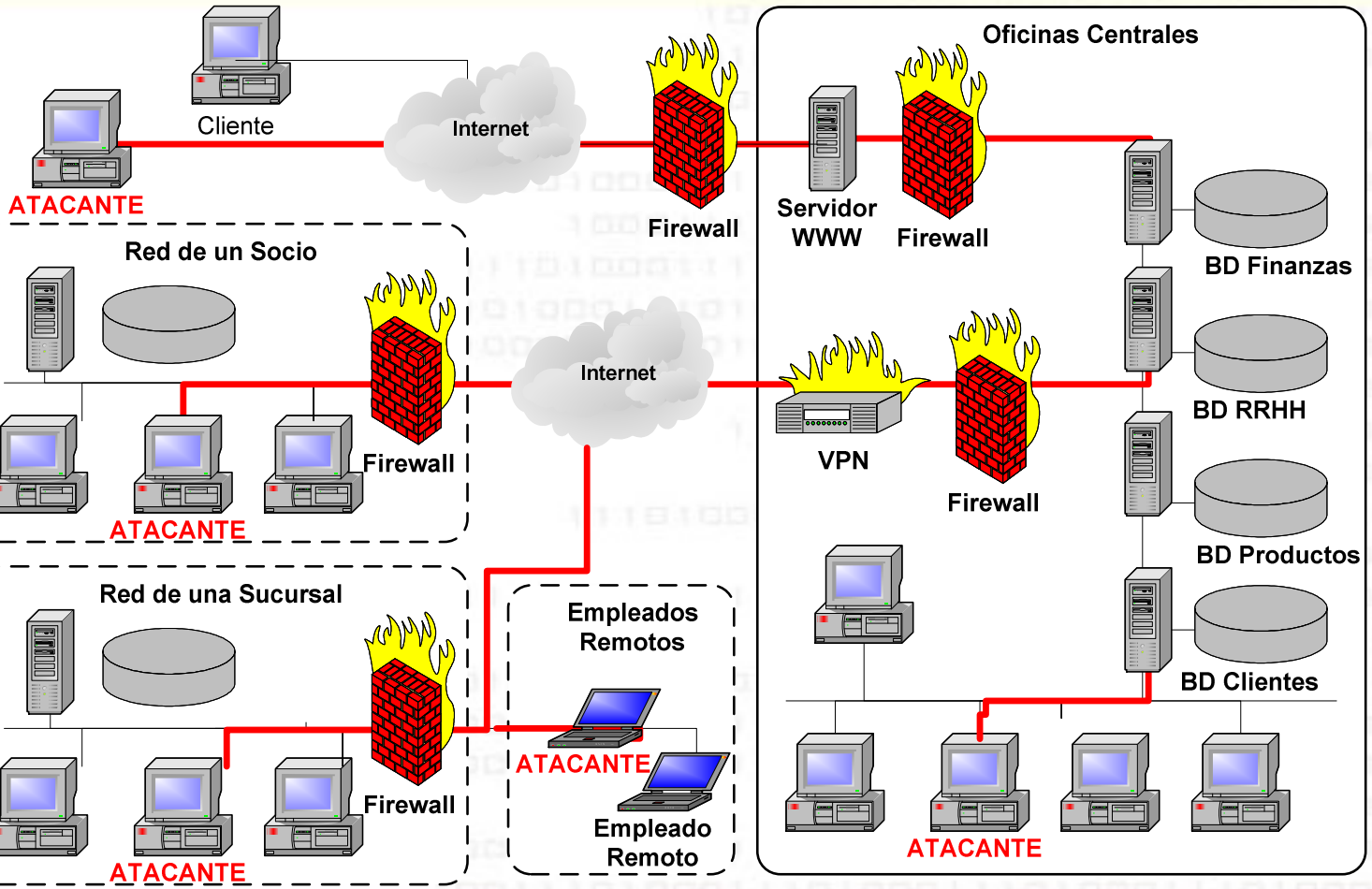
Evolución de la seguridad en Bases de datos

- Antes (1990s)
  - Esquema más centralizado, con servidores más aislados
  - Las BDs eran exclusivamente para uso interno
  - Problemas de seguridad muy poco conocidos
- Ahora
  - BDs accesibles al público desde Internet
  - BDs compartidas con proveedores, clientes y socios.
- Las Bases de datos son más útiles cuando su información está disponible a más personas
  - Esto aumenta la amenaza a su seguridad
- Se necesitan proteger todos los niveles

Introducción a la Seguridad en Aplicaciones de bases datos

- La mayoría de las empresas tiene alguna aplicación web disponible para Internet y/o para uso interno.
- Comúnmente obtienen información de un servidor de base de datos.
- Estas bases de datos parecen estar seguras debido a que los usuarios no están directamente conectados a ellas, sino a través del servidor web.
- Importancia de la seguridad en las Bases de Datos
  - Las Bases de datos contienen la información más valiosa en una Empresa.
  - Es el principal blanco de ataques.

Amenazas a las Bases de datos



Vulnerabilidades de SQL Injection

- Es uno de los problemas de seguridad más comunes en las aplicaciones de bases de datos.
- Es un error de Programación que puede pasar inadvertido (la aplicación funciona correctamente).
- Se puede ser vulnerable cuando se arma una instrucción SQL concatenando variables que ingresa el usuario.

Vulnerabilidades de SQL Injection

- El usuario puede manipular los parámetros de forma que logre ejecutar una instrucción SQL distinta.
- La vulnerabilidad puede estar en:
  - Una aplicación del usuario
  - Procedimientos almacenados: Definidos por el usuario o que son parte de la base de datos.
- Todos los motores de bases de datos están afectados
  - Cada uno tiene distintas características que pueden hacer más fácil o difícil un ataque.

Vulnerabilidades de SQL Injection en Aplicaciones Oracle

- Hay dos tipos de bloque SQL donde puede haber SQL Injection:
  - Bloque SQL anónimo
    - Un Bloque SQL que tiene un BEGIN y un END y que puede ejecutar más de una sentencia SQL.
    - No hay limitación a lo que el atacante puede hacer. Se permiten instrucciones SELECT, DML y DDL.
  - Sentencia SQL simple
    - No tiene un BEGIN y un END.
    - El atacante no puede insertar ";" para inyectar más comandos SQL.

Vulnerabilidades de SQL Injection en Aplicaciones Oracle

- Bloque SQL anónimo - Ejemplo código vulnerable (ASP):

```
conn.Execute "BEGIN  CREATE_NEW_EMP (" & Request.Form("EMPNO") & ",
'" & Request.Form("EMPNAME") &"');  END;"
```

  - Uso normal: EMPNO = 7020 y EMPNAME = JUAN PEREZ
  - Comando resultante:

```
BEGIN
  CREATE_NEW_EMP (7020, 'JUAN PEREZ');
END;
```

  - Ataque usando SQL Injection: EMPNO = 7021 y EMPNAME = J HACKER'); UPDATE SCOTT.EMP SET SAL = 70000 WHERE EMPNO = 7021; COMMIT; END;--
  - Comando resultante:

```
BEGIN
  CREATE_NEW_EMP (7021, 'J HACKER');
  UPDATE SCOTT.EMP SET SAL = 70000 WHERE EMPNO = 7021;
  COMMIT;
END;--');  END;"
```

Vulnerabilidades de SQL Injection en Aplicaciones Oracle

- Sentencia SQL simple - Ejemplo código vulnerable (ASP):

```
rs.Open "SELECT EMPNO, ENAME, JOB FROM SCOTT.EMP WHERE
ENAME LIKE '" & Request.Form("Search") & "%'", conn,
3, 3
```

  - Uso normal: Search = PEREZ
  - Consulta resultante:

```
SELECT EMPNO, ENAME, JOB FROM SCOTT.EMP WHERE ENAME
LIKE 'PEREZ%'
```

  - Ataque usando SQL Injection:
    Search = ' || LLAMADA_A_FUNCION() ||'
  - Consulta resultante:

```
SELECT EMPNO, ENAME, JOB FROM SCOTT.EMP WHERE ENAME
LIKE ' ' || LLAMADA_A_FUNCION() || ' %'
```

Inyectando una llamada a función

- Características que debe tener la función:
  - Debe crear una transacción autónoma
    - Necesario debido a que sino se obtiene un error ya que se ejecuta dentro de una consulta.
- DBMS_XMLQUERY.GETXML
  - Esta función ejecuta una consulta y devuelve el resultado en formato XML.
- Permite ejecutar bloques PL/SQL anónimos como el usuario actual
  - Se puede crear una transacción autónoma.
  - Ejecutar no solo consultas sino también sentencias DML y DDL.
  - No existe una elevación de privilegios, pero esto se puede utilizar para explotar fácilmente las vulnerabilidades SQL Injection que requieren una función creada.

Utilizando UNION en un ataque de SQL Injection

- Se puede agregar un UNION y una consulta SELECT para realizar robo de información de la base de datos.
- Se debe respetar la misma cantidad de campos y tipo de datos que la consulta original de la aplicación.
  - Prueba y error.
- Un ataque usando el ejemplo anterior sería:

```
Search = ' UNION SELECT deptno,dname,loc,0 FROM scott.dept--
```

- La consulta SQL resultante que ejecuta la página web es:

```
SELECT EMPNO, ENAME, JOB FROM SCOTT.EMP WHERE ENAME LIKE ''
UNION SELECT deptno,dname,loc,0 FROM scott.dept--%'
```

- Este exploit obtiene información de la base de datos distinta para lo que originalmente se la había programado.

Herramienta de Demostración - SQLIT

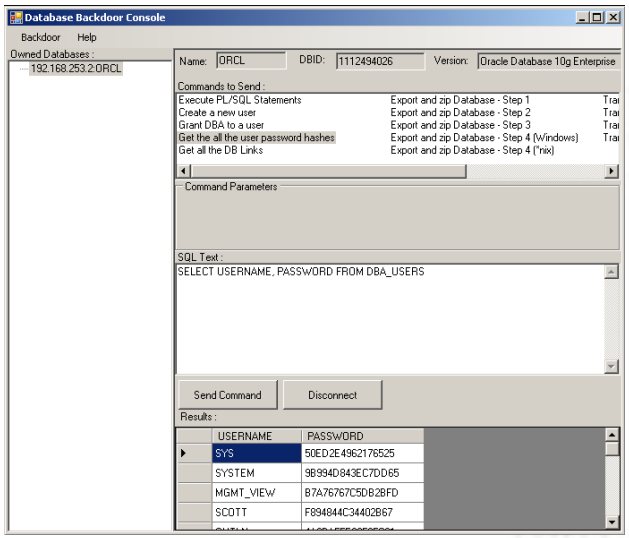
- Captura los parámetros de las distintas páginas de un sitio web.
- Permite probar exploits de SQL Injection en los distintos parámetros
- Tiene cargados varios exploits.
- Permite especificar los comandos que se desean ejecutar.

Script para obtener control remoto de un servidor Oracle - Backdoor

- Abre una conexión saliente a un host y puerto determinados.
- **Importante**: No modifica en nada la base de datos.
- Consola de Administración del Backdoor
  - Una aplicación con interfaz gráfica (GUI) que:
    - Envía comandos al Backdoor instalado en la Base de datos y recibe la salida.
    - Muestra la información acerca de el o los Backdoors desplegados.
    - Configura el Backdoor.
    - Permite la administración de múltiples Backdoors.

Script para obtener control remoto de un servidor Oracle - Backdoor

Consola del Backdoor



Host del atacante (remoto)

- Escucha en un puerto TCP

Oracle Database Server

- Envía info. de la BD adueñada
- Muestra la nueva BD adueñada
- Envía comando
- Ejecuta el comando
- Envía la salida del comando
- Muestra la salida

Repetir hasta que se recibe “EXIT”

Elevación de privilegios utilizando SQL Injection dentro de la base de datos

- Si el usuario de la aplicación no tiene privilegios suficientes, el atacante puede aprovechar una vulnerabilidad dentro de la base de datos para elevar privilegios.
- Vulnerabilidad de SQL Injection en DBMS_CDC_ISUBSCRIBE.SUBSCRIBE
  - Arreglada en el parche crítico emitido por Oracle en Abril de 2005.
  - Permite inyectar la llamada a una función que va a ser ejecutada como el usuario SYS.

Combinando ataques de SQL Injection

- Se pueden combinar varios ataques de SQL Injection para llevar a cabo un ataque más complejo.
- SQL Injection en Employees.asp
  - Permite inyectar la llamada a una función que va a ejecutar como el usuario de la aplicación.
- SQL Injection in DBMS_CDC_ISUBSCRIBE.SUBSCRIBE
  - Permite ejecutar una función con privilegios de SYS.
- DBMS_XMLQUERY.GETXML
  - Función que permite ejecutar un programa PL/SQL como el usuario actual (Usuario de aplicación).

- Resultado: Combinando todos estos ataques se logra ejecutar comandos con privilegios administrativos en la base de datos utilizando un sitio web.

¿Cómo protegerse? - Evitando vulnerabilidades en la aplicación

- Previniendo vulnerabilidades de SQL Injection:
  - Utilizar objetos Command con parámetros.
  - Utilizar procedimientos almacenados.
  - Si se va a armar una sentencia SQL dinámicamente: Validar cuidadosamente las variables utilizadas.
    - Validación de las entradas:
    - En campos de texto: Reemplazar las comillas simples (') por dobles comillas simples ('').
    - En campos numéricos: Asegurarnos que existen solo números y no hay texto.
- Capacitar adecuadamente a los programadores sobre los riesgos y la forma de evitar las vulnerabilidades.
- Asegurarnos que no estamos devolviendo información de error al usuario web.

¿Cómo protegerse?

- Utilizar un IDS/IPS que detecte ataques a nivel de aplicación.
- El usuario de la aplicación debe tener los mínimos privilegios necesarios. Nunca debe tener privilegios de DBA.
- Mantener el Software (Base de datos y SO) con los parches al día .

Conclusiones

- Las aplicaciones web pueden ser un punto de acceso vulnerable a la información de una empresa.
  - Suelen estar accesibles a una gran cantidad de personas (Internet).
- Importancia en la capacitación de los programadores de las aplicaciones para que estén concientes de los riesgos y saber cómo evitarlos.
- Se debe asegurar la base de datos y el sistema operativo por más que la misma esté detrás de un firewall y solo el usuario de la aplicación web y/o usuarios internos de la empresa tengan acceso a la base de datos.
  - Si se encuentra un punto vulnerable en alguna aplicación es más fácil elevar privilegios.
  - El usuario interno puede ser una amenaza.

Referencias

- SQL Injection
  - http://www.securitydocs.com/library/3587
- SQL Injection en Oracle
  - http://www.securityfocus.com/infocus/1644
  - http://www.securityfocus.com/infocus/1646
  - http://security-papers.globint.com.ar/oracle_security/sql_injection_in_oracle.php
- Oracle Backdoors
  - https://www.blackhat.com/presentations/bh-europe-07/Cerrudo/Whitepaper/bh-eu-07-cerrudo-WP-up.pdf
  - https://www.blackhat.com/presentations/bh-europe-07/Cerrudo/bh-eu-07-Cerrudo-Tools.zip
- Vulnerabilidades en Bases de Datos
  - http://www.argeniss.com/research.html
  - http://www.appsecinc.com/resources/alerts/oracle/

END

- ¿Preguntas?
- Muchas Gracias.
- Contacto: esteban>at<argeniss>dot<com