Lab 5. Aplicațiile pe baza de date si securitatea datelor

Cuvinte cheie: • Contextul aplicației • SQL dinamic	• Drepturile creatorului • Drepturile apelantului • SQL injection

I. Contextul aplicației

Contextul aplicației este asemănător unei variabile globale de tip record dintr-un pachet PL/SQL, insa valoarea ei nu poate fi modificata de către utilizator prin simpla atribuire. Odată setata la login, valoarea contextului aplicației poate fi citita pe întreaga durata a sesiunii utilizatorului. Valoarea acestei variabile poate fi schimbata doar printr-un apel de procedura.

Totodată aceasta variabila este menținuta de Oracle in PGA(Program Global Area) si astfel este privata fiecărei sesiuni utilizator.

Aşa cum am văzut in laboratorul trecut, exista moduri de a da sau nu drept de execuție asupra unei proceduri stocate către utilizatorii de rând.

Pasii de configurare a unui context de aplicație de către *SYS AS SYSDBA*:
1) Crearea contextului :

```
CREATE CONTEXT aplicatie_ctx USING proced_aplicatie_ctx;
```

2) Crearea procedurii asociate contextului, proced_aplicatie_ctx.

In mod implicit un utilizator are un context implicit USERENV , care conține o mulțime de atribute de context precum:

- IP_ADDRESS, adresa IP. In cazul serverului, va fi blank la interogare;
- AUTHENTICATION_TYPE, l-am folosit in laboratorul 1 ca sa vedem tipul autentificării pentru utilizatorii creați;
- CURRENT_SQL, l-am folosit in laboratorul 1 ca sa auditam ce instructiuni SQL folosea utilizatorul;
- LANGUAGE, limba sesiunii utilizator ;s.a.

Administratorul bazei de date poate defini in mod analog atribute pentru contextul APLICATIE_CTX, prin execuția procedurii asociate contextului , proced_aplicatie_ctx.
De exemplu, se utilizează atributul contextual LANG_RO.

Sa ne propunem ca exemplu sa nu permitem nimănui, nici măcar administratorului de aplicație, sa modifice sau sa insereze date de nume / prenume de utilizator daca nu are limba sesiunii setata la limba romana.

```
CREATE OR REPLACE PROCEDURE proced_aplicatie_ctx IS
V_LANG VARCHAR(50);

BEGIN

   SELECT SYS_CONTEXT ('USERENV', 'LANGUAGE') INTO V_LANG FROM DUAL;

   DBMS_OUTPUT.PUT_LINE('LANGUAGE:'||V_LANG);
   IF V_LANG NOT LIKE '%ROMANIAN%' THEN
        BEGIN
         DBMS_OUTPUT.PUT_LINE('PENTRU CA NU VA ESTE SETATA LIMBA ROMANA
NU O SA AVETI VOIE SA MODIFICATI/INSERATI IN CAMPUL DE NUME SI PRENUME
UTILIZATOR');
            DBMS_SESSION.set_context ('APLICATIE_CTX', 'LANG_RO', 'NU');
           END;
   ELSE
           DBMS_SESSION.set_context ('APLICATIE_CTX', 'LANG_RO', 'DA');

   END IF;
END;
/
```

Când executa procedura SYS AS SYSDBA ii afiseaza informaţiile:

```
EXEC proced_aplicatie_ctx();
```

```
SQL> EXEC proced_aplicatie_ctx();
LANGUAGE:AMERICAN_AMERICA.WE8MSWIN1252
PENTRU CA NU VA ESTE SETATA LIMBA ROMANA NU O SA AVETI VOIE SA
MODIFICATI/INSERATI IN CAMPUL DE NUME SI PRENUME UTILIZATOR

PL/SQL procedure successfully completed.
```

4) --> Pas realizat de catre ELEARN_APP_ADMIN
In triggerul de insert pe tabela ELEARN_APP_ADMIN.UTILIZATOR, verificam atributul contextual pentru a impune restricţiile aplicaţiei:

Utilizatorul ELEARN_APP_ADMIN creează triggerul:

```
CREATE OR REPLACE TRIGGER TR_INSERT_USER
BEFORE INSERT ON ELEARN_APP_ADMIN.UTILIZATOR
FOR EACH ROW
DECLARE
v_poate VARCHAR2(4);
BEGIN
 v_poate:=SYS_CONTEXT ('APLICATIE_CTX', 'LANG_RO');
 IF v_poate='NU' THEN
   DBMS_OUTPUT.PUT_LINE('NU AVETI VOIE SA INSERATI NUME/PRENUME FARA
DIACRITICE!');
   :NEW.NUME:=NULL;
   :NEW.PRENUME:=NULL;
 END IF;
END;
/
```

Analog pentru triggerul de update pe înregistrările aceleiaşi tabele UTILIZATOR:

```
CREATE OR REPLACE TRIGGER TR_UPDATE_USER
BEFORE UPDATE ON ELEARN_APP_ADMIN.UTILIZATOR
FOR EACH ROW
DECLARE
v_poate VARCHAR2(4);
BEGIN
  v_poate:=SYS_CONTEXT ('APLICATIE_CTX', 'LANG_RO');
  IF v_poate='NU' THEN
    DBMS_OUTPUT.PUT_LINE('NU AVETI VOIE SA INSERATI NUME/PRENUME FARA
DIACRITICE!');
    :NEW.NUME:=NULL;
    :NEW.PRENUME:=NULL;
  END IF;
END;
/
```

Apoi utilizatorul ELEARN_APP_ADMIN isi încheie sesiunea cu exit;

5) SYS AS SYSDBA creeaza un trigger de logon care la conectarea la baza de date, automat se setează contextul aplicaţiei pentru sesiunea utilizatorului:

```
CREATE OR REPLACE TRIGGER TR_AFTER_LOGON
AFTER LOGON
ON DATABASE
BEGIN
  proced_aplicatie_ctx();
END;
/
```

6) Se reconecteaza utilizatorul ELEARN_APP_ADMIN si incearca urmatorul insert:

```
INSERT INTO UTILIZATOR
VALUES(3,'STUDENT','ZIMNEA','VICTOR','ELEARN_student4',SYSDATE-600,NULL);
```

Este clar ca daca limba sesiunii este setata la limba engleza atunci nu va fi scris corect.

```
Enter user-name: ELEARN_APP_ADMIN
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SET SERVEROUTPUT ON
SQL> DESC UTILIZATOR
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(4)
 TIP                                                VARCHAR2(12)
 NUME                                      NOT NULL VARCHAR2(20)
 PRENUME                                   NOT NULL VARCHAR2(20)
 NUMEUSER                                  NOT NULL VARCHAR2(20)
 AN_INTRARE                                NOT NULL DATE
 AN_IESIRE                                          DATE

SQL> SELECT * FROM UTILIZATOR
  2  ;

        ID TIP          NUME                 PRENUME
---------- ------------ -------------------- --------------------
NUMEUSER             AN_INTRAR AN_IESIRE
-------------------- --------- ---------
         1 STUDENT      ANTON                SAUL
ELEARN_student2      30-OCT-11

         2 STUDENT      ARSENIE              SANDRA
ELEARN_student3      13-MAY-09


SQL> INSERT INTO UTILIZATOR VALUES(3,'STUDENT','CRIVAT','VICTOR','ELEARN_student4',SYSDATE-600,NULL);
NU AVETI VOIE SA INSERATI NUME/PRENUME FARA DIACRITICE!
INSERT INTO UTILIZATOR VALUES(3,'STUDENT','CRIVAT','VICTOR','ELEARN_student4',SYSDATE-600,NULL)
*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("ELEARN_APP_ADMIN"."UTILIZATOR"."NUME")


SQL> _
```

Un utilizator supărat de aceasta restricţie nu va putea modifica direct valoarea atributului contextual. Daca totuşi va încerca, va primi următoarea eroare:

BEGIN
DBMS_SESSION.set_context ('APLICATIE_CTX', 'LANG_RO', 'DA');

END;
/

```
SQL> BEGIN
  2    DBMS_SESSION.set_context ('APLICATIE_CTX', 'LANG_RO', 'DA');
  3
  4  END;
  5  /
BEGIN
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 101
ORA-06512: at line 2
```

Reţinem ca modificarea atributelor contextuale se poate realiza doar de către procedura ataşata contextului. In acest caz prin execuţia procedurii **proced_aplicatie_ctx**

II. SQL Dinamic si riscurile de securitate pe care le prezintă

SQL dinamic reprezintă acele instrucțiuni care permit execuția oricărui cod SQL la runtime. Ne-am întâlnit cu acest concept la cursul de SGBD din anul 3, când am studiat cursoarele dinamice.

II.1.Cursoare dinamice

Sa recapitulam cum arata un **cursor dinamic**

ELEARN_APP_ADMIN creaza o procedura cu cursor dinamic care este gandita a fi folosita pentru a obține informații complete din tabela REZOLVA.

```
CREATE OR REPLACE PROCEDURE PROC_CURSOR_DINAM(cerere_sql VARCHAR2) AS

   TYPE tip_ref_c IS REF CURSOR;
   ref_c tip_ref_c;
   v_rand ELEARN_APP_ADMIN.REZOLVA%ROWTYPE;
BEGIN
   OPEN ref_c FOR cerere_sql;
   LOOP
      FETCH ref_c INTO v_rand;
      EXIT WHEN ref_c%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('STUDENTUL:'||v_rand.ID_STUD||' LA TEMA:'||v_rand.ID_TEMA||' ARE
NOTA:'||NVL(v_rand.NOTA,0));

   END LOOP;
   CLOSE ref_c;


END;
/
```

ELEARN_APP_ADMIN acorda privilegiu de execuție a procedurii către profesorul ELEARN_profesor1:

```
GRANT EXECUTE ON PROC_CURSOR_DINAM TO ELEARN_profesor1;
```

Profesorul ELEARN_profesor1 executa procedura corect prima data:

```
EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM
ELEARN_APP_ADMIN.REZOLVA');
```

```
SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM ELEARN_APP_ADMIN.REZOLVA');
STUDENTUL:2 LA TEMA:1 ARE NOTA:10
STUDENTUL:1 LA TEMA:2 ARE NOTA:9

PL/SQL procedure successfully completed.
```

Apoi, printr-o cerere modificata reuseste sa pătrundă si la informații confidențiale, ce țin strict de secretariat, precum informația daca respectivul student este la reluare de studii:

```
EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT R.* FROM ELEARN_APP_ADMIN.REZOLVA R, ELEARN_APP_ADMIN.CURSANT C WHERE R.ID_STUD=C.ID AND C.RELUARE_STUDII=1 ');
```

Acest exemplu arata in ce mod cursoarele dinamice dau acces de vizualizare la informații confidențiale, daca nu sunt bine administrate.

Reținem insa ca riscul este doar de a divulga informații, nu si de a opera modificări. Aceasta deoarece prin cursoare dinamice se pot realiza numai interogari.

Daca s-ar încerca o comanda LMD sau LDD aceasta ar eşua, de exemplu:

```
EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DELETE FROM ELEARN_APP_ADMIN.REZOLVA');
```

```
SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DELETE FROM ELEARN_APP_ADMIN.REZOLVA');
BEGIN ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DELETE FROM ELEARN_APP_ADMIN.REZOLVA'); END;

*
ERROR at line 1:
ORA-06539: target of OPEN must be a query
ORA-06512: at "ELEARN_APP_ADMIN.PROC_CURSOR_DINAM", line 7
ORA-06512: at line 1
```

II.2. EXECUTE IMMEDIATE

O alta forma pe care o putem întâlni pentru SQL dinamic este următoarea, bazata pe **EXECUTE IMMEDIATE**. Aceasta este printre cele mai vulnerabile metode de a executa cod dinamic, întrucât poate permite si acțiuni modificatoare de date.

Utilizam acelaşi exemplu de mai sus si recreem procedura cu un nou conținut:

```
CREATE OR REPLACE PROCEDURE PROC_CURSOR_DINAM(cerere_sql VARCHAR2)
AS
TYPE vector IS TABLE OF  ELEARN_APP_ADMIN.REZOLVA%ROWTYPE;
v_vector vector;

BEGIN
  EXECUTE IMMEDIATE(cerere_sql) BULK COLLECT INTO  v_vector;
  FOR i IN 1..v_vector.COUNT LOOP
     DBMS_OUTPUT.PUT_LINE('STUDENTUL:'||v_vector(i).ID_STUD||'                LA TEMA:'||v_vector(i).ID_TEMA||' ARE NOTA:'||NVL(v_vector(i).NOTA,0));
  END LOOP;

END;
/

GRANT EXECUTE ON PROC_CURSOR_DINAM TO ELEARN_profesor1;
```

Profesorul ELEARN_profesor1 executa procedura corect prima data:

EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM ELEARN_APP_ADMIN.REZOLVA');

```
SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM ELEARN_APP_ADMIN.REZOLVA');
STUDENTUL:2 LA TEMA:1 ARE NOTA:10
STUDENTUL:1 LA TEMA:2 ARE NOTA:9

PL/SQL procedure successfully completed.
```

Profesorul ELEARN_profesor1 executa procedura modificata a doua data si reuseste sa stearga toate înregistrările din tabela REZOLVA :

EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DECLARE v_id NUMBER(4); BEGIN DELETE FROM ELEARN_APP_ADMIN.REZOLVA;COMMIT; SELECT id_stud INTO v_id FROM ELEARN_APP_ADMIN.REZOLVA WHERE ID_STUD=1 AND ID_TEMA=2; END; ');

```
SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DECLARE v_id NUMBER(4); BEGIN DELETE FROM ELEARN_APP_ADMIN.REZOLVA;COMMI
T id_stud INTO v_id FROM ELEARN_APP_ADMIN.REZOLVA WHERE ID_STUD=1 AND ID_TEMA=2;  END; ');

PL/SQL procedure successfully completed.

SQL> SELECT * FROM ELEARN_APP_ADMIN.REZOLVA;

no rows selected
```

Ce a făcut de fapt?

A inclus un întreg bloc PL/SQL ca argument pentru comanda EXECUTE IMMEDIATE.

Cum a reuşit acest lucru, din punct de vedere al privilegiilor?

Sa vedem ce privilegii are in sesiunea curenta utilizatorul ELEARN_profesor1:

select * from session_privs;

```
SQL> select * from session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
```

Intr-adevăr, putem sa interogam si ca SYS AS SYSDBA, vedem ca nu profesorul nu are privilegii de ştergere pe tabela

SELECT substr(grantee,1,15) grantee, owner, substr(table_name,1,15) table_name, grantor,privilege FROM DBA_TAB_PRIVS WHERE grantee='ELEARN_profesor1';

```
SQL> SELECT substr(grantee,1,15) grantee, owner, substr(table_name,1,15) table_name, grantor,privilege FROM DBA_TAB_PRIV
E grantee='ELEARN_profesor1';

no rows selected
```

Al treilea mod de a ne convinge este sa ca profesorul sa încerce comanda de ştergere direct din prompt:

```
SQL> delete from ELEARN_APP_ADMIN.REZOLVA;
delete from ELEARN_APP_ADMIN.REZOLVA
                            *
ERROR at line 1:
ORA-01031: insufficient privileges
```

***Deci apelantul* ELEARN_profesor1 *nu ar fi avut drept sa stearga din tabela* ELEARN_APP_ADMIN.REZOLVA.**
Dar a reusit deoarece a executat procedura in contextul de privilegii al creatorului procedurii, adica al lui ELEARN_APP_ADMIN.

Ne reamintim schema din laboratorul trecut.

userul X creeaza un obiect de tip view⟨trigger, procedura⟩

	in schema proprie			*in schema altui user Y*		
	acceseaza obiecte din schema proprie	*acceseaza obiecte din schema lui Y* (select Y.D, insert Y.D)		*acceseaza obiecte din schema proprie*	*acceseaza obiecte din schema lui Y* (select Y.D, insert Y.D)	
Ce privilegii are nevoie X	*CREATE VIEW*	*CREATE VIEW*		*CREATE ANY VIEW*	*CREATE ANY VIEW*	
		SELECT ON Y.D *INSERT ON Y.D*	*SELECT ON Y.D WITH GRANT OPTION* *INSERT ON Y.D WITH GRANT OPTION*		*SELECT ON Y.D* *INSERT ON Y.D*	*SELECT ON Y.D WITH GRANT OPTION* *INSERT ON Y.D WITH GRANT OPTION*
Ce privilegii are nevoie apelantul Z	*SELECT ON VIZ* *INSERT ON VIZ*	*SELECT ON VIZ* *INSERT ON VIZ* *SELECT ON Y.D* *INSERT ON Y.D*	*SELECT ON VIZ* *INSERT ON VIZ*	*SELECT ON VIZ* *INSERT ON VIZ*	*SELECT ON VIZ* *INSERT ON VIZ* *SELECT ON Y.D* *INSERT ON Y.D*	*SELECT ON VIZ* *INSERT ON VIZ*

Ne aflam in cazul in care userul X (ELEARN_APP_ADMIN) a creat o procedura in schema proprie, care accesează obiecte din schema proprie (ELEARN_APP_ADMIN.REZOLVA).

Prin urmare, apelantul Z (ELEARN_profesor1) are nevoie doar de privilegii asupra procedurii (ELEARN_APP_ADMIN. PROC_CURSOR_DINAM) ca sa poata executa cu aceasta orice i se permite si creatorului procedurii .

Cum ne protejam de astfel de atacuri?

Prima modalitate este prin adăugarea clauzei AUTHID CURRENT_USER in antetul procedurii, se va folosi la runtime doar contextul de privilegii al apelantului.
Aceasta tehnica este denumita in engleza „Invoker Rights' Model".

```
CREATE OR REPLACE PROCEDURE PROC_CURSOR_DINAM(cerere_sql VARCHAR2)
AUTHID CURRENT_USER
AS
TYPE vector IS TABLE OF  ELEARN_APP_ADMIN.REZOLVA%ROWTYPE;
v_vector vector;

BEGIN
   EXECUTE IMMEDIATE(cerere_sql) BULK COLLECT INTO  v_vector;
   FOR i IN 1..v_vector.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE('STUDENTUL:'||v_vector(i).ID_STUD||' LA
TEMA:'||v_vector(i).ID_TEMA||' ARE NOTA:'||NVL(v_vector(i).NOTA,0));
   END LOOP;

END;
/

GRANT EXECUTE ON PROC_CURSOR_DINAM TO ELEARN_profesor1;
```

Testam efectul. Mai întâi ELEARN_APP_ADMIN reface datele tabelei REZOLVA prin următoarele insert-uri:

```
INSERT INTO REZOLVA (ID_TEMA,ID_STUD,DATA_UPLOAD) VALUES(1,2,SYSDATE-3);
INSERT INTO REZOLVA (ID_TEMA,ID_STUD,DATA_UPLOAD) VALUES(2,1,SYSDATE-7);
COMMIT;
```

Acum utilizatorul **ELEARN_profesor1** *încearcă din nou sa execute cele doua apeluri, unul cu select si unul cu bloc PL/SQL inclus:*

```
EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM
ELEARN_APP_ADMIN.REZOLVA');

EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DECLARE v_id NUMBER(4); BEGIN
DELETE FROM ELEARN_APP_ADMIN.REZOLVA;COMMIT; SELECT id_stud INTO v_id
FROM ELEARN_APP_ADMIN.REZOLVA WHERE ID_STUD=1 AND ID_TEMA=2;  END; ');
```

```
SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM ELEARN_APP_ADMIN.REZOLVA');
STUDENTUL:2 LA TEMA:1 ARE NOTA:0
STUDENTUL:1 LA TEMA:2 ARE NOTA:0

PL/SQL procedure successfully completed.

SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DECLARE v_id NUMBER(4); BEGIN DELETE FROM ELEARN_APP_ADMIN.REZOLVA;COMMIT;
I id_stud INTO v_id FROM ELEARN_APP_ADMIN.REZOLVA WHERE ID_STUD=1 AND ID_TEMA=2;  END; ')
BEGIN ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DECLARE v_id NUMBER(4); BEGIN DELETE FROM ELEARN_APP_ADMIN.REZOLVA;COMMIT; SEL
_stud INTO v_id FROM ELEARN_APP_ADMIN.REZOLVA WHERE ID_STUD=1 AND ID_TEMA=2;  END; '); END;

*
ERROR at line 1:
ORA-06550: line 1, column 60:
PL/SQL: ORA-01031: insufficient privileges
ORA-06550: line 1, column 31:
PL/SQL: SQL Statement ignored
ORA-06512: at "ELEARN_APP_ADMIN.PROC_CURSOR_DINAM", line 8
ORA-06512: at line 1
```

Acum, al doilea apel, cel cu codul PL/SQL malitios nu a mai reusit, deoarece procedura a fost executata cu drepturile apelantului, care nu avea privilegiu de stergere pe tabela ELEARN_APP_ADMIN.REZOLVA **.**

A doua modalitate de a ne proteja de vulnerabilitatile codului SQL dinamic este prin utilizarea expresiilor regulate. Astfel, vom face o validare a cererii introduse de utilizator inainte de a o executa.

Pentru exemplul nostru anterior, se rescrie procedura astfel, ca sa testam ca si in cazul in care ar avea privilegii LMD pe tabela, utilizatorul va folosi procedura doar pentru interogari:

```
CREATE OR REPLACE PROCEDURE PROC_CURSOR_DINAM(cerere_sql VARCHAR2)
AUTHID CURRENT_USER
AS
TYPE vector IS TABLE OF  ELEARN_APP_ADMIN.REZOLVA%ROWTYPE;
v_vector vector;
este_ok NUMBER(1) :=0;
BEGIN

   IF REGEXP_LIKE(cerere_sql,'SELECT [A-Za-z0-9*]+ [^;]') THEN
        este_ok:=1;
   END IF;

   IF este_ok = 1 THEN BEGIN
          EXECUTE IMMEDIATE(cerere_sql) BULK COLLECT INTO  v_vector;
          FOR i IN 1..v_vector.COUNT LOOP
             DBMS_OUTPUT.PUT_LINE('STUDENTUL:'||v_vector(i).ID_STUD||'                    LA
TEMA:'||v_vector(i).ID_TEMA||' ARE NOTA:'||NVL(v_vector(i).NOTA,0));
          END LOOP;
   END;
   ELSE
      DBMS_OUTPUT.PUT_LINE('Comanda contine cod suspect a fi malitios. Sunt permise doar
interogari de date');
   END IF;

END;
/

GRANT EXECUTE ON PROC_CURSOR_DINAM TO ELEARN_profesor1;
```

Sa presupunem ca pentru acest exemplu strict i se acorda si privilegiul DELETE pe tabela REZOLVA profesorului.

```
GRANT DELETE ON ELEARN_APP_ADMIN.REZOLVA TO ELEARN_profesor1;
```

Acum utilizatorul **ELEARN_profesor1** *încearcă din nou sa execute cele doua apeluri, unul cu select si unul cu bloc PL/SQL inclus:*

```
EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM
ELEARN_APP_ADMIN.REZOLVA');

EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DECLARE v_id NUMBER(4); BEGIN
DELETE FROM ELEARN_APP_ADMIN.REZOLVA;COMMIT; SELECT id_stud INTO v_id
FROM ELEARN_APP_ADMIN.REZOLVA WHERE ID_STUD=1 AND ID_TEMA=2;  END; ');
```

```
SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM ELEARN_APP_ADMIN.REZOLVA');
STUDENTUL:2 LA TEMA:1 ARE NOTA:0
STUDENTUL:1 LA TEMA:2 ARE NOTA:0

PL/SQL procedure successfully completed.

SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('DECLARE v_id NUMBER(4); BEGIN DELETE FROM ELEARN_APP_ADMIN.REZOLVA;COMMI
T id_stud INTO v_id FROM ELEARN_APP_ADMIN.REZOLVA WHERE ID_STUD=1 AND ID_TEMA=2;  END; ');
Comanda contine cod suspect a fi malitios. Sunt permise doar interogari de date

PL/SQL procedure successfully completed.

SQL> SELECT * FROM ELEARN_APP_ADMIN.REZOLVA;

   ID_TEMA    ID_STUD DATA_UPLO       NOTA ID_CORECTOR
---------- ---------- --------- ---------- -----------
         1          2 25-AUG-12
         2          1 21-AUG-12
```

Observam din captura de ecran efectul verificării cu expresii regulate. Încercarea utilizatorului de a executa cod malițios a eşuat.

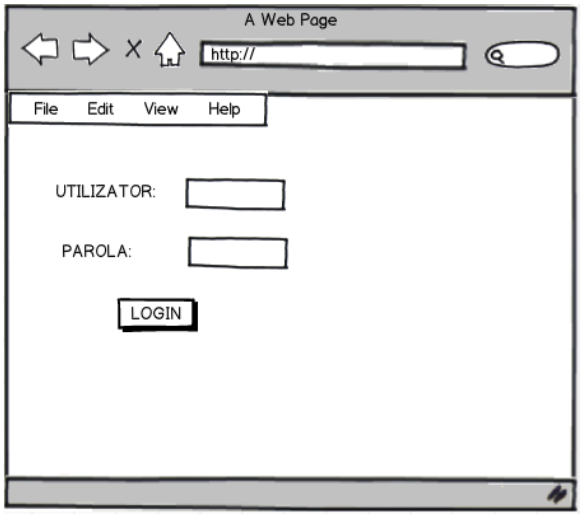
Pentru a restabili privilegiile initiale, ii revocam profesorului dreptul de a şterge înregistrări pe tabela REZOLVA , dat strict pentru acest exemplu didactic:

REVOKE DELETE ON ELEARN_APP_ADMIN.REZOLVA FROM ELEARN_profesor1;

III. SQL injection

SQL injection este un tip de atac informatic care presupune introducerea de fragmente de cod cu sintaxa SQL valida in cadrul unei cereri SQL existente, cu scop de a produce efecte distructive.

Un exemplu tipic este o cerere care in clauza WHERE foloseşte valori primite la runtime. De cele mai multe ori ținta atacurilor de SQL injection sunt formularele de login neprotejate la astfel de atacuri.



Pentru a putea rezolva acest exercițiu, utilizatorul *ELEARN_APP_ADMIN adauga la tabela UTILIZATOR o coloana care va stoca parole in forma criptata.*

ALTER TABLE UTILIZATOR ADD PAROLA VARCHAR2(32);

Pentru testare, populam cu dummy data acest câmp, refolosind cunostintele din laboratorul 1, despre criptare:

[SYS AS SYSDBA trebuie sa ii dea privilegiul de a utiliza pe deplin pachetul de criptare userului ELEARN_APP_ADMIN:

GRANT ALL ON DBMS_CRYPTO TO ELEARN_APP_ADMIN;

]

Functia CRIPTARE1 cripteaza un sir primit ca parametru folosind algoritmul DES, cheia '12345678', padding cu zero-uri si metoda de chaining ECB.

```
create or replace FUNCTION criptare1(textclar IN VARCHAR2) RETURN VARCHAR2 AS
 raw_sir RAW(20);
 raw_parola RAW(20);
 rezultat RAW(20);
 parola VARCHAR2(20) := '12345678';
 mod_operare NUMBER;
 textcriptat VARCHAR2(32);
BEGIN
  raw_sir:=utl_i18n.string_to_raw(textclar,'AL32UTF8');
  raw_parola :=utl_i18n.string_to_raw(parola,'AL32UTF8');
  mod_operare := DBMS_CRYPTO.ENCRYPT_DES + DBMS_CRYPTO.PAD_ZERO +
DBMS_CRYPTO.CHAIN_ECB;
  rezultat := DBMS_CRYPTO.ENCRYPT(raw_sir,mod_operare,raw_parola);
  --dbms_output.put_line(rezultat);
  textcriptat := RAWTOHEX (rezultat);
  RETURN textcriptat;

END;
/
```

In continuare, administratorul aplicației ELEARN actualizează parolele din tabela UTILIZATOR:

```
UPDATE UTILIZATOR SET PAROLA=criptare1('Parola1') WHERE ID=1;
UPDATE UTILIZATOR SET PAROLA=criptare1('Parola2') WHERE ID=2;

SET LINESIZE 200
SELECT * FROM UTILIZATOR;
```

```
SELECT * FROM UTILIZATOR;

   ID TIP          NUME                 PRENUME              NUMEUSER             AN_INTRAR AN_IESIRE PAROLA
----- ------------ -------------------- -------------------- -------------------- --------- --------- ----------------
--------
    1 STUDENT      ANTON                SAUL                 ELEARN_student2      30-OCT-11           699B9532C48C3497
    2 STUDENT      ARSENIE              SANDRA               ELEARN_student3      13-MAY-09           959B8BB0E2267E14
```

Acum putem continua discuția despre SQL injection.

Sa presupunem ca butonului de „LOGIN” de pe formular ii corespunde o procedura stocata creata de administratorul aplicației de e-learning, care are ca scop verificarea daca exista potrivire intre numele de utilizator si parola introduse prin formular.

```
CREATE OR REPLACE PROCEDURE VERIFICA_LOGIN (P_NUMEUSER
VARCHAR2,P_PAROLA VARCHAR2) AS
v_ok NUMBER(2) :=-1;
BEGIN
  EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM UTILIZATOR WHERE
NUMEUSER='''||P_NUMEUSER||''' AND PAROLA=criptare1('''||P_PAROLA||''')' INTO v_ok;
  dbms_output.put_line('SELECT COUNT(*) FROM UTILIZATOR WHERE
NUMEUSER='''||P_NUMEUSER||''' AND PAROLA=criptare1('''||P_PAROLA||''')');
  IF v_ok=0 THEN
    DBMS_OUTPUT.PUT_LINE('VERIFICAREA A ESUAT');

  ELSE
    DBMS_OUTPUT.PUT_LINE('VERIFICAREA A REUSIT');
  END IF;
END;
/
```

Ce fel de atacuri malițioase s-ar putea întâmpla prin apelul acestei proceduri cu parametrii rău intentionati?

Valoarea parametrului **P_NUMEUSER**	Valoarea parametrului **P_PAROLA**	Rezultatul
'ELEARN_student2'	'Parola1'	**EXEC VERIFICA_LOGIN('ELEARN_student2','Parola1');** SQL> EXEC VERIFICA_LOGIN('ELEARN_student2','Parola1'); SELECT COUNT(*) FROM UTILIZATOR WHERE NUMEUSER='ELEARN_student2' AND PAROLA=criptare1('Parola1') VERIFICAREA A REUSIT
'ELEARN_student2'	'Parola2'	**EXEC VERIFICA_LOGIN('ELEARN_student2','Parola2');** SQL> EXEC VERIFICA_LOGIN('ELEARN_student2','Parola2'); SELECT COUNT(*) FROM UTILIZATOR WHERE NUMEUSER='ELEARN_student2' AND PAROLA=criptare1('Parola2') VERIFICAREA A ESUAT PL/SQL procedure successfully completed.
'ELEARN_student2**''--**'	'Parola2'	**EXEC VERIFICA_LOGIN('ELEARN_student2''--','Parola2');** SQL> EXEC VERIFICA_LOGIN('ELEARN_student2''--','Parola2'); SELECT COUNT(*) FROM UTILIZATOR WHERE NUMEUSER='ELEARN_student2'--' AND PAROLA=criptare1('Parola2') VERIFICAREA A REUSIT PL/SQL procedure successfully completed. In acest caz, *persoana malițioasa cunoaşte un nume valid de utilizator, dar nu-i cunoaşte parola*. Prin adăugarea apostroafelor si a celor 2 liniuțe inhiba (drept comentariu) partea de parola din cererea select.
'ABRACADABRA99**'' OR 1=1 --**'	'HOCUS-POCUS'	**EXEC VERIFICA_LOGIN('ABRACADABRA99'' OR 1=1 --','HOCUS-POCUS');** SQL> EXEC VERIFICA_LOGIN('ABRACADABRA99'' OR 1=1 --','HOCUS-POCUS'); SELECT COUNT(*) FROM UTILIZATOR WHERE NUMEUSER='ABRACADABRA99' OR 1=1 --' AND PAROLA=criptare1('HOCUS-POCUS') VERIFICAREA A REUSIT PL/SQL procedure successfully completed. In acest caz, *persoana malițioasa nu cunoaşte nici macar un nume valid de utilizator.* Prin adăugarea apostroafelor si a celor 2 liniuțe inhiba (drept comentariu) partea de parola din cererea select. Mai mul, clauza OR permite in acest caz anularea si a testului de existenta a numelui de utilizator in tabela.

Cum ne protejam de astfel de atacuri?

Prin înlocuirea concatenării din şirul ce reprezintă comanda SQL cu variabile legate (engl. bind variables)
In cazul procedurii de verificare a corespondentei nume utilizator – parola in formular:

Varianta 1 de rescriere pentru protectie:

```
CREATE OR REPLACE PROCEDURE VERIFICA_LOGIN_SAFE (P_NUMEUSER
VARCHAR2,P_PAROLA VARCHAR2) AS
v_ok NUMBER(2) :=-1;
BEGIN
  SELECT COUNT(*) INTO v_ok FROM UTILIZATOR WHERE NUMEUSER=P_NUMEUSER
AND PAROLA=criptare1(P_PAROLA);
  IF v_ok=0 THEN
    DBMS_OUTPUT.PUT_LINE('VERIFICAREA A ESUAT');

  ELSE
    DBMS_OUTPUT.PUT_LINE('VERIFICAREA A REUSIT');
  END IF;
END;
/
```

La apel, acum se furnizează parametrii după cum urmează:

```
ACCEPT NUME PROMPT 'NUME UTILIZATOR:'

ACCEPT PAROL PROMPT 'PAROLA DVS:'

EXEC VERIFICA_LOGIN_SAFE ('&NUME','&PAROL');
```

Reluam testele pentru a ne asigura ca acum nu mai au succes atacurile malițioase:

Valoarea parametrului **P_NUMEUSER**	Valoarea parametrului **P_PAROLA**	Rezultatul
'ELEARN_student2'	'Parola1'	SQL> ACCEPT NUME PROMPT 'NUME UTILIZATOR:' NUME UTILIZATOR:ELEARN_student2 SQL> ACCEPT PAROL PROMPT 'PAROLA DVS:' PAROLA DVS:Parola1 SQL> EXEC VERIFICA_LOGIN_SAFE ('&NUME','&PAROL'); VERIFICAREA A REUSIT PL/SQL procedure successfully completed.
'ELEARN_student2'	'Parola2'	SQL> ACCEPT NUME PROMPT 'NUME UTILIZATOR:' NUME UTILIZATOR:ELEARN_student2 SQL> ACCEPT PAROL PROMPT 'PAROLA DVS:' PAROLA DVS:Parola2 SQL> EXEC VERIFICA_LOGIN_SAFE ('&NUME','&PAROL'); VERIFICAREA A ESUAT PL/SQL procedure successfully completed.
'ELEARN_student2''-- '	'Parola2'	SQL> ACCEPT NUME PROMPT 'NUME UTILIZATOR:' NUME UTILIZATOR:ELEARN_student2''-- SQL> ACCEPT PAROL PROMPT 'PAROLA DVS:' PAROLA DVS:Parola2 SQL> EXEC VERIFICA_LOGIN_SAFE ('&NUME','&PAROL'); VERIFICAREA A ESUAT PL/SQL procedure successfully completed.
'ABRACADABRA99'' OR 1=1 --'	'HOCUS-POCUS'	SQL> ACCEPT NUME PROMPT 'NUME UTILIZATOR:' NUME UTILIZATOR:ABRACADABRA99'' OR 1=1 -- SQL> ACCEPT PAROL PROMPT 'PAROLA DVS:' PAROLA DVS:HOCUS-POCUS SQL> EXEC VERIFICA_LOGIN_SAFE ('&NUME','&PAROL'); VERIFICAREA A ESUAT PL/SQL procedure successfully completed.

Varianta 2 de rescriere pentru protectie:

```
CREATE OR REPLACE PROCEDURE VERIFICA_LOGIN_SAFE2 (P_NUMEUSER
VARCHAR2,P_PAROLA VARCHAR2) AS
v_ok NUMBER(2) :=-1;
BEGIN
  EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM UTILIZATOR WHERE
NUMEUSER=:numeus AND PAROLA=criptare1(:parol)' INTO v_ok USING
P_NUMEUSER,P_PAROLA;
  IF v_ok=0 THEN
    DBMS_OUTPUT.PUT_LINE('VERIFICAREA A ESUAT');

  ELSE
    DBMS_OUTPUT.PUT_LINE('VERIFICAREA A REUSIT');
  END IF;
END;
/
```

La apel, acum se furnizează parametrii după cum urmează:

```
EXEC VERIFICA_LOGIN_SAFE2('ELEARN_student2','Parola1');
EXEC VERIFICA_LOGIN_SAFE2('ELEARN_student2','Parola2');
EXEC VERIFICA_LOGIN_SAFE2('ELEARN_student2''--','Parola2');
EXEC VERIFICA_LOGIN_SAFE2('ABRACADABRA99'' OR 1=1 --','HOCUS-POCUS');
```

Reluam testele pentru a ne asigura ca acum nu mai au succes atacurile malițioase:

```
SQL> EXEC VERIFICA_LOGIN_SAFE2('ELEARN_student2','Parola1');
VERIFICAREA A REUSIT

PL/SQL procedure successfully completed.

SQL> EXEC VERIFICA_LOGIN_SAFE2('ELEARN_student2','Parola2');
VERIFICAREA A ESUAT

PL/SQL procedure successfully completed.

SQL> EXEC VERIFICA_LOGIN_SAFE2('ELEARN_student2''--','Parola2');
VERIFICAREA A ESUAT

PL/SQL procedure successfully completed.

SQL> EXEC VERIFICA_LOGIN_SAFE2('ABRACADABRA99'' OR 1=1 --','HOCUS-POCUS');
VERIFICAREA A ESUAT

PL/SQL procedure successfully completed.
```

La final, ştergem coloana parola, pentru a nu influenta rezolvările ulterioare:
ALTER TABLE UTILIZATOR DROP COLUMN PAROLA;

Exercitii

1. Creați un context de aplicație care sa stabilească ca si măsura de securitate posibilitatea de evaluare a temelor depuse de studenți de către cadre didactice doar in intervalul orar de la facultate, orele 8.00-20.00.

2. Creați o procedura sub o sesiune a utilizatorului ***ELEARN_APP_ADMIN*** , care sa execute imediat orice cerere primeşte ca input si sa returneze numărul înregistrărilor rezultat al cererii primite la runtime. Apelantul va fi userul ELEARN_profesor1.

a) Executati procedura corect, cu un select, cu drepturile creatorului
b) Executati procedura in mod malitios, cu drepturile creatorului
c) Executati procedura corect, cu un select, cu drepturile apelantului
d) Executati procedura in mod malitios, cu drepturile apelantului.

3. Gasiti toate breşele de securitate in următoarea procedura, al carei scop era sa afiseze temele tuturor studenților depuse intr-un an sau intr-o luna sau intr-o data exacta furnizata ca parametru de intrare:

```
CREATE OR REPLACE PROCEDURE GASITI_PERICOLE(P_DATAUP VARCHAR2) AS
TYPE vector IS TABLE OF  ELEARN_APP_ADMIN.REZOLVA%ROWTYPE;
v_vector vector;

BEGIN

  EXECUTE IMMEDIATE 'SELECT * FROM REZOLVA WHERE
TO_CHAR(DATA_UPLOAD,''DD-MM-YYYY HH24:MI:SS'') LIKE ''%'||P_DATAUP||'%''
BULK COLLECT INTO  v_vector;
  FOR i IN 1..v_vector.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE('STUDENTUL:'||v_vector(i).ID_STUD||' LA
TEMA:'||v_vector(i).ID_TEMA||'UPLOADATA LA DATA: '||v_vector(i).DATA_UPLOAD||' ARE
NOTA:'||NVL(v_vector(i).NOTA,-1));
  END LOOP;


END;
/
```

Sugestii de posibile atacuri malițioase:
1) Obțineți si toate informațiile despre utilizatorii aplicației, de când sunt ei in sistem
2) Obțineți si informații suplimentare despre statutul studenților, care din ei sunt la reluare de studii

Resurse suplimentare:
[1] http://st-curriculum.oracle.com/tutorial/SQLInjection/index.htm