Securitatea bazelor de date – master anul 2
Laborator 5

Aplicaţiile pe baza de date și securitatea datelor

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

1. Contextul aplicaţiei

- **Contextul aplicației** este asemănător unei variabile globale de tip record dintr-un pachet PL/SQL, însă valoarea ei nu poate fi modificată de către utilizator prin simpla atribuire.
- Odată setată la login, valoarea contextului aplicaţiei poate fi citită pe întreaga durata a sesiunii utilizatorului. Valoarea acestei variabile poate fi schimbată doar printr-un apel de procedură.
- Totodată, această variabilă este menţinută de Oracle în PGA (*Program Global Area*) și astfel este privată fiecărei sesiuni utilizator.
- Aşa cum am văzut în laboratorul trecut, există moduri de a da sau nu drept de execuţie asupra unei proceduri stocate către utilizatorii de rând.

- În mod implicit un utilizator are contextul (implicit) *USERENV*, care conţine o mulţime de atribute de context precum:
  - *IP_ADDRESS* – adresa IP. În cazul serverului, va fi *blank* la interogare;
  - *AUTHENTICATION_TYPE* – l-am folosit în laboratorul 3 ca să găsim tipul autentificării pentru utilizatorii creaţi;
  - *CURRENT_SQL* – l-am folosit în laboratorul 2 ca să auditam instrucțiunile SQL executate de către utilizator;
  - *LANGUAGE* – limba sesiunii utilizator ; ș.a.
- Administratorul bazei de date poate defini în mod analog atribute pentru contextul pe care îl creează, prin execuția procedurii asociate contextului.

- Pașii de configurare a unui context de aplicaţie de către *SYS AS SYSDBA*:
  1. **Crearea contextului:**

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

  2. **Crearea procedurii asociate contextului** (*proced_aplicatie_ctx*)**.**

     ```
     CREATE OR REPLACE PROCEDURE proced_aplicatie_ctx IS
     …
     BEGIN
     …
     DBMS_SESSION.set_context ('APLICATIE_CTX', 'ATRIBUT_NOU',
     ```

```
'VALOARE');
…
END;
/
```

Reținem că modificarea atributelor contextuale se poate realiza doar de către procedura ataşată contextului. În cazul exemplului de mai sus, modificarea atributelor contextului se poate realiza prin execuția procedurii proced_aplicatie_ctx. Pentru aceasta, SYS AS SYSDBA execută comanda:

```
EXEC proced_aplicatie_ctx();
```

3. SYS AS SYSDBA creează un ***trigger* de logon** care determină ca, la conectarea la baza de date, să fie stabilit automat contextul aplicației pentru sesiunea utilizatorului:

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

4. Ulterior, ***în trigger*-i BEFORE INSERT / BEFORE UPDATE** se va putea testa valoarea returnată de apelul *SYS_CONTEXT*('*APLICATIE_CTX*', '*ATRIBUT_NOU*') și, în funcţie de aceasta, se va permite sau nu operaţia LMD.

- **Exemplu:** Exerciţiul 1 propus (și rezolvat) la finalul laboratorului.

2. SQL Dinamic și riscurile de securitate pe care le prezintă

- SQL dinamic reprezintă acele instrucţiuni care permit execuţia oricărui cod SQL la *runtime*. (Conceptul a fost prezentat la cursul de SGBD.)

2.1 Cursoare dinamice

- Cum arată un cursor dinamic?
- **Exemplu:** ELEARN_App_Admin creează o procedură cu cursor dinamic al cărei scop este de a fi folosită pentru a obţine lista studenţilor din sistem.

```
--SYS AS SYSDBA executa comenzile:
create user elearn_app_admin identified by 12345;
grant connect to elearn_app_admin;
alter user elearn_app_admin quota 2m on users;
grant create table to elearn_app_admin;
grant create procedure to elearn_app_admin;

--elearn_app_admin:
create table student(id number primary key,
```

```
        nume varchar2(30), prenume varchar2(30), anul number,
        specializare varchar2(3), grupa number);
create table materie(id number primary key,
                    denumire varchar2(20));
create table examen (id number primary key, cod_materie number,
             dataex date,
             constraint fk_ex2 foreign key(cod_materie)
             references materie(id));
create table evaluare (cod_student number not null,
        cod_examen number not null,nota number(4,2) default -1,
        constraint pk_ev1 primary key (cod_student,cod_examen),
        constraint fk_ev1 foreign key (cod_student) references
        student(id), constraint fk_ex1 foreign key (cod_examen)
        references examen(id));

insert into student values (1,'A','Abc',2,'Inf',231);
insert into student values(2,'B','Bbc',2,'Inf',231);

insert into materie values(1,'Algebra');

insert into examen values(1,1,sysdate-700);
insert into examen values(2,1,sysdate-300);

insert into evaluare values(1,1,3);
insert into evaluare values(2,1,10);
insert into evaluare values(1,2,9);

--user-ul elearn_asistent3 încearcă să selecteze date direct din
tabelele lui elearn_app_admin:
SELECT EV.cod_student||EV.nota||EX.DATAEX||MAT.DENUMIRE Info
FROM elearn_app_admin.evaluare ev,
     elearn_app_admin.examen ex, elearn_admin.materie mat
WHERE ev.cod_examen=ex.id
AND ex.cod_materie=mat.id;
```

==> **Eroare:** insufficient privileges

```
--acum elearn_app_admin creează o procedură cu cursor dinamic:
CREATE OR REPLACE PROCEDURE PROC_CDINAM(cerere_sql VARCHAR2) AS
  TYPE tip_ref_c IS REF CURSOR;
  ref_c tip_ref_c;
  v_sir VARCHAR2(200);
BEGIN
  OPEN ref_c FOR cerere_sql;
  LOOP
     FETCH ref_c INTO v_sir;
     EXIT WHEN ref_c%NOTFOUND;
```

```
        DBMS_OUTPUT.PUT_LINE('STUDENTUL: '||v_sir);
    END LOOP;
    CLOSE ref_c;
END;
/

-- și acordă privilegiu de execuție a procedurii de către
asistentul elearn_asistent3:
grant execute on proc_cdinam to elearn_asistent3;

--elearn_asistent3 revine
SELECT EV.cod_student||EV.nota||EX.DATAEX||MAT.DENUMIRE Info
FROM elearn_admin.evaluare ev,
     elearn_admin.examen ex, elearn_admin.materie mat
WHERE ev.cod_examen=ex.id
AND ex.cod_materie=mat.id;
```

==> **Eroare:** insufficient privileges

```
--Încearcă să folosească procedura cu cursor dinamic:
set serveroutput on;
exec elearn_admin.proc_cdinam('select nume from student');

exec elearn_admin.proc_cdinam('SELECT EV.cod_student||
EV.nota||EX.DATAEX||MAT.DENUMIRE Info FROM elearn_admin.evaluare
ev, elearn_admin.examen ex,elearn_admin.materie mat WHERE
ev.cod_examen=ex.id AND ex.cod_materie=mat.id');
```

```
SQL> set serveroutput on;
SQL> exec elearn_admin.proc_cdinam('select nume from student');
STUDENTUL:A
STUDENTUL:B

PL/SQL procedure successfully completed.

SQL> exec elearn_admin.proc_cdinam('select EV.cod_student||EV.nota||EX.DATAEX||MAT.DENUMIRE sir from elearn_ad
elearn_admin.examen ex,elearn_admin.materie mat where ev.cod_examen=ex.id and ex.cod_materie=mat.id');
STUDENTUL:1312-JAN-11Algebra
STUDENTUL:21012-JAN-11Algebra
STUDENTUL:1916-FEB-12Algebra

PL/SQL procedure successfully completed.
```

- Acest exemplu arată în ce mod cursoarele dinamice oferă acces de vizualizare la informaţii confidenţiale, dacă nu sunt bine administrate.
- **Reținem**, însă, că riscul este doar de a divulga informaţii, nu și de a opera modificări. Acest lucru se întâmplă deoarece prin cursoare dinamice se pot realiza numai interogări.
- Dacă s-ar încerca o comandă LMD sau LDD aceasta ar eşua, de exemplu:

```
exec elearn_admin.proc_cdinam('delete from evaluare');
```

```
SQL> exec elearn_admin.proc_cdinam('delete from evaluare');
BEGIN elearn_admin.proc_cdinam('delete from evaluare'); END;

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

2.2 *EXECUTE IMMEDIATE*

- O altă formă pe care o putem întâlni pentru SQL dinamic este următoarea, bazată pe instrucțiunea *EXECUTE IMMEDIATE*. Aceasta este printre cele mai vulnerabile metode de a executa cod dinamic, întrucât poate permite și acţiuni modificatoare de date.
- Utilizăm acelaşi exemplu de mai sus și recreem procedura cu un nou conţinut:

```
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* execută procedura corect prima dată:

```
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, profesorul *ELEARN_profesor1* execută procedura modificată a doua dată și reușește să șteargă toate înregistrările din tabelul *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?
  - Verificăm ce privilegii are în sesiunea curentă utilizatorul *ELEARN_profesor1*:

```
select * from session_privs;
SQL> select * from session_privs;

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

* Putem să interogăm și ca *SYS AS SYSDBA* pentru a verifica. Într-adevăr, vom observa că profesorul nu are privilegii de ştergere pe tabel:

```
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 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
```

* **Observăm** că apelantul *ELEARN_profesor1* nu ar fi avut dreptul să șteargă din tabelul *ELEARN_APP_ADMIN.REZOLVA*. **Însă,** a reușit deoarece a executat procedura **în contextul de privilegii al creatorului procedurii**, adică 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 aflăm în cazul în care userul *X* (*ELEARN_APP_ADMIN*) a creat o procedură în schema proprie, iar aceasta accesează obiecte din schema proprie (tabelul

ELEARN_APP_ADMIN.REZOLVA).

- Prin urmare, apelantul *Z* (*ELEARN_profesor1*) are nevoie doar de privilegii asupra procedurii (*ELEARN_APP_ADMIN. PROC_CURSOR_DINAM*) ca să poată executa cu aceasta orice i se permite creatorului procedurii.

Cum ne protejăm de astfel de atacuri?

- **Prima modalitate** este prin **adăugarea clauzei *AUTHID CURRENT_USER*** în antetul procedurii. Astfel, se va folosi la *runtime* doar contextul de privilegii al apelantului. Aceasta tehnică este denumită în engleză „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;
```

- Testăm efectul. Mai întâi ELEARN_APP_ADMIN reface datele tabelului REZOLVA prin următoarele comenzi *insert*:

```
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;
```

- Utilizatorul *ELEARN_profesor1* încearcă din nou să execute cele două apeluri, unul cu select și 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
```

- Observăm că al doilea apel (cel cu codul PL/SQL malițios) nu a mai reușit, deoarece procedura a fost executată cu drepturile apelantului, care nu avea privilegiu de ștergere pe tabelul *ELEARN_APP_ADMIN.REZOLVA*.

- **A doua modalitate** de a ne proteja de vulnerabilitățile codului SQL dinamic este prin utilizarea expresiilor regulate. Astfel, vom face o validare a cererii introduse de utilizator înainte de a o executa.
- Pentru exemplul nostru anterior, se rescrie procedura astfel, pentru a testa că și în cazul în care ar avea privilegii LMD pe tabel, utilizatorul va folosi procedura doar pentru interogări:

```
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;
```

- Să presupunem că pentru acest exemplu strict i se acordă profesorului și privilegiul *DELETE* pe tabelul *REZOLVA*.

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

- Utilizatorul *ELEARN_profesor1* încearcă din nou să execute cele două apeluri, unul cu *SELECT* și 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; ');
```

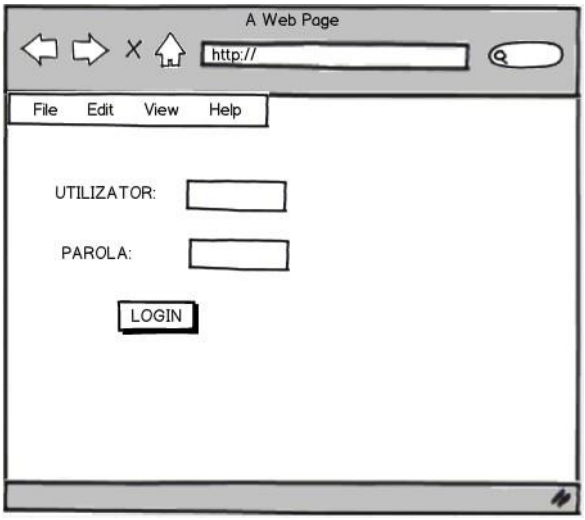
```
SQL> EXEC ELEARN_APP_ADMIN.PROC_CURSOR_DINAM('SELECT * FROM ELEARN_APP_ADMIN.REZOLVA');
STUDENTUL:2 LA TEMA:1 ARE NOTA:0
STUDENTUL:1 LA TEMA:2 ARE NOTA:0

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
```

- Observăm din captura de ecran efectul verificării cu expresii regulate. Încercarea utilizatorului de a executa cod malițios a eșuat.
- Pentru a restabili privilegiile inițiale, îi revocăm profesorului dreptul de a șterge înregistrări pe tabelul *REZOLVA* , dat strict pentru acest exemplu didactic:

```
REVOKE DELETE ON ELEARN_APP_ADMIN.REZOLVA FROM ELEARN_profesor1;
```

3. SQL injection

- SQL injection este un tip de atac informatic care presupune introducerea de fragmente de cod cu sintaxă SQL validă în cadrul unei cereri SQL existente, având potențial de a produce efecte distructive.
- Un exemplu tipic este o cerere care în clauza *WHERE* folosește valori primite la *runtime*. De multe ori ținta atacurilor de SQL injection sunt formularele de *login* neprotejate la astfel de atacuri.



- Pentru a putea exemplifica, utilizatorul *ELEARN_APP_ADMIN adaugă la tabelul UTILIZATOR o coloană care va stoca parole în forma criptată.*

```
ALTER TABLE UTILIZATOR ADD PAROLA VARCHAR2(32);
```

- Pentru testare, populăm cu *dummy data* acest câmp, refolosind cunoștințele din laboratorul 1 (despre criptare):

```
--SYS AS SYSDBA trebuie să îi 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 criptează un șir primit ca parametru folosind
algoritmul DES, cheia '12345678', padding cu zero-uri și 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;
/
--În continuare, administratorul aplicaţiei (ELEARN_APP_ADMIN)
actualizează parolele din tabelul 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
```

- Putem continua discuţia despre SQL Injection. Să presupunem că butonului de „LOGIN” de pe formular îi corespunde o procedură stocată creată de administratorul aplicaţiei de *e-learning*, care are ca scop verificarea potrivirii între numele de utilizator și 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 parametri rău intenționați?

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. În acest caz, *persoana malițioasă cunoaşte un nume valid de utilizator, dar nu-i cunoaşte parola*. Prin adăugarea apostrofurilor și a celor 2 liniuţe inhibă (drept comentariu) partea de parolă 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. În acest caz, *persoana malițioasă nu cunoaşte nici măcar un nume valid de utilizator*. Prin adăugarea apostrofurilor și a celor 2 liniuţe inhibă (drept comentariu) partea de parolă din cererea SELECT. Mai mult, clauza OR permite în acest caz și anularea testului de existență a numelui de utilizator în tabel.

Cum ne protejăm de astfel de atacuri?

- Prin înlocuirea concatenării din şirul ce reprezintă comanda SQL cu variabile de legătură (*bind variables*)
- Prezentăm variantele de protecție în cazul procedurii de verificare a corespondenței nume utilizator – parolă în formular.
- **Varianta 1** de rescriere pentru protecție:

```
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, se furnizează parametrii după cum urmează:
ACCEPT NUME PROMPT 'NUME UTILIZATOR:' ACCEPT PAROL PROMPT 'PAROLA
DVS:'
EXEC VERIFICA_LOGIN_SAFE ('&NUME','&PAROL');
```

- Reluăm testele pentru a ne asigura că 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 protecție:

```
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, 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');
```

- Reluăm testele pentru a ne asigura că 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 influența rezolvările ulterioare:

```
ALTER TABLE UTILIZATOR DROP COLUMN PAROLA;
```

4. Exerciții

1. **Creaţi un context de aplicaţie care stabilește, ca măsură de securitate, posibilitatea de evaluare a temelor depuse de studenţi de către cadre didactice doar în intervalul orar de la facultate, orele 8.00-20.00.**

2. **Găsiți toate breşele de securitate din următoarea procedură, al cărei scop era să afişeze temele tuturor studenţilor depuse într-un an sau într-o lună sau la o dată exactă furnizată 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 maliţioase:

1. Obţineţi și toate informaţiile despre utilizatorii aplicaţiei, de când sunt ei în sistem.
2. Obţineţi și informaţii suplimentare despre statutul studenţilor (care din ei sunt la reluare de studii).