Project - Database Security

Ciprian-Mihai Ceausescu

ciprian-mihai.ceausescu@my.fmi.unibuc.ro

University of Bucharest — November 24, 2018

Introduction

In this project is presented a database model for a company that can sells **products** to **users** and, also, saves a history of all **orders**. The entities are the following:

- user - the main entity contains the following information: **authentication** e-mail, password, reset password token and timestamp, created and updated timestamps.
- role - contains the following information: **name** of the role, created and updated timestamps.
- product - contains the following information: **name** of the product, description, price, image url, created and updated timestamps.
- order - contains the following information: **pay type**, how the product is delivered, created and updated timestamps.

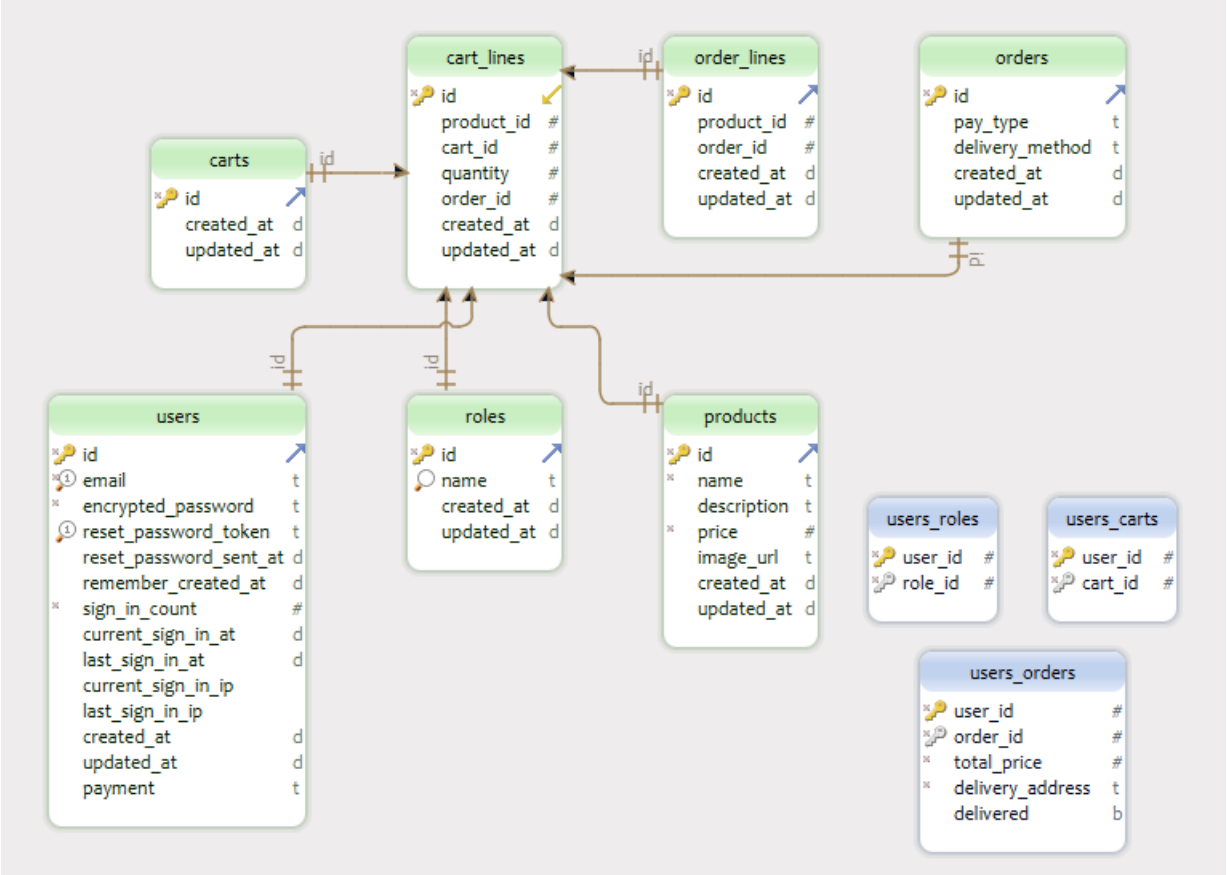

Figure 1: Main database diagram

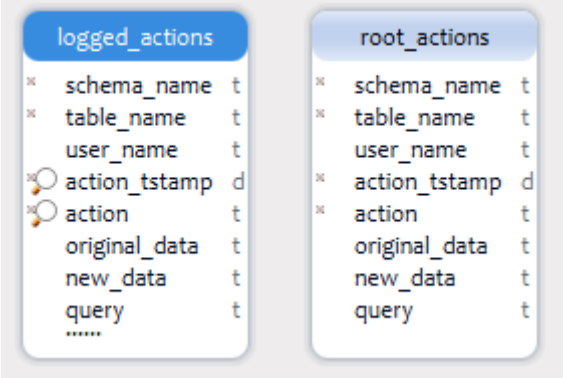

Figure 2: Audit database diagram

```
-- Database: database_security

-- DROP DATABASE database_security;

CREATE DATABASE database_security
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'English_United States.1252'
    LC_CTYPE = 'English_United States.1252'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

COMMENT ON DATABASE database_security
    IS 'Database Security Project - Ciprian Mihai Ceausescu';
```

Figure 3: Database and tables - Database

```
CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg_catalog;
COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';

CREATE EXTENSION pgcrypto;

SET search_path = public, pg_catalog;
SET default_tablespace = '';
SET default_with_oids = false;
```

Figure 4: Database and tables - Configurations

```
CREATE TABLE users (
    id integer NOT NULL,
    email character varying DEFAULT ''::character varying NOT NULL,
    encrypted_password character varying DEFAULT ''::character varying NOT NULL,
    reset_password_token character varying,
    reset_password_sent_at timestamp without time zone,
    remember_created_at timestamp without time zone,
    sign_in_count integer DEFAULT 0 NOT NULL,
    current_sign_in_at timestamp without time zone,
    last_sign_in_at timestamp without time zone,
    current_sign_in_ip inet,
    last_sign_in_ip inet,
    created_at timestamp without time zone,
    updated_at timestamp without time zone
);
```

Figure 5: Database and tables - Users

```
ALTER TABLE ONLY users ADD CONSTRAINT users_pkey PRIMARY KEY (id);
CREATE SEQUENCE users_id_seq START WITH 1 INCREMENT BY 1 NO MINVALUE NO MAXVALUE CACHE 1;
ALTER SEQUENCE users_id_seq OWNED BY users.id;
ALTER TABLE ONLY users ALTER COLUMN id SET DEFAULT nextval('users_id_seq'::regclass);
CREATE UNIQUE INDEX index_users_on_email ON users USING btree (email);
CREATE UNIQUE INDEX index_users_on_reset_password_token ON users USING btree (reset_password_token);
```

Figure 6: Database and tables - Users

```
CREATE TABLE roles (
    id integer NOT NULL,
    name character varying,
    created_at timestamp without time zone,
    updated_at timestamp without time zone
);
ALTER TABLE ONLY roles ADD CONSTRAINT roles_pkey PRIMARY KEY (id);
CREATE SEQUENCE roles_id_seq START WITH 1 INCREMENT BY 1 NO MINVALUE NO MAXVALUE CACHE 1;
ALTER SEQUENCE roles_id_seq OWNED BY roles.id;
ALTER TABLE ONLY roles ALTER COLUMN id SET DEFAULT nextval('roles_id_seq'::regclass);
CREATE INDEX index_roles_on_name ON roles USING btree (name);
```

Figure 7: Database and tables - Roles

```
CREATE TABLE users_roles (
    user_id integer,
    role_id integer
);
ALTER TABLE users_roles ADD CONSTRAINT compose_pk PRIMARY KEY(user_id, role_id);
CREATE INDEX index_users_roles_on_user_id_and_role_id ON users_roles USING btree (user_id, role_id);
```

Figure 8: Database and tables - Users Roles

```
CREATE TABLE products (
    id integer NOT NULL,
    name character varying NOT NULL,
    description character varying,
    price double precision NOT NULL,
    image_url character varying,
    created_at timestamp without time zone,
    updated_at timestamp without time zone
);
ALTER TABLE ONLY products ADD CONSTRAINT products_pkey PRIMARY KEY (id);
CREATE SEQUENCE products_id_seq START WITH 1 INCREMENT BY 1 NO MINVALUE NO MAXVALUE CACHE 1;
ALTER SEQUENCE products_id_seq OWNED BY products.id;
ALTER TABLE ONLY products ALTER COLUMN id SET DEFAULT nextval('products_id_seq'::regclass);
```

Figure 9: Database and tables - Products

```
CREATE TABLE carts (
    id integer NOT NULL,
    created_at timestamp without time zone,
    updated_at timestamp without time zone
);
ALTER TABLE ONLY carts ADD CONSTRAINT carts_pkey PRIMARY KEY (id);
CREATE SEQUENCE carts_id_seq START WITH 1 INCREMENT BY 1 NO MINVALUE NO MAXVALUE CACHE 1;
ALTER SEQUENCE carts_id_seq OWNED BY carts.id;
ALTER TABLE ONLY carts ALTER COLUMN id SET DEFAULT nextval('carts_id_seq'::regclass);
```

Figure 10: Database and tables - Carts

```
CREATE TABLE users_carts (
   user_id integer,
   cart_id integer
);
ALTER TABLE users_carts ADD CONSTRAINT compose_pk_users_carts PRIMARY KEY(user_id, cart_id);
CREATE INDEX index_users_carts_on_user_id_and_cart_id ON users_carts USING btree (user_id, cart_id);
```

Figure 11: Database and tables - Users Carts

```
CREATE TABLE orders (
    id integer NOT NULL,
    pay_type text,
    delivery_method text,
    created_at timestamp without time zone,
    updated_at timestamp without time zone
);
ALTER TABLE ONLY orders ADD CONSTRAINT orders_pkey PRIMARY KEY (id);
CREATE SEQUENCE orders_id_seq START WITH 1 INCREMENT BY 1 NO MINVALUE NO MAXVALUE CACHE 1;
ALTER SEQUENCE orders_id_seq OWNED BY orders.id;
ALTER TABLE ONLY orders ALTER COLUMN id SET DEFAULT nextval('orders_id_seq'::regclass);
```

Figure 12: Database and tables - Orders

```
CREATE TABLE users_orders (
    user_id integer NOT NULL,
    order_id integer NOT NULL,
    total_price double precision NOT NULL,
    delivery_address text NOT NULL,
    delivered boolean default FALSE,
    created_at timestamp without time zone,
    updated_at timestamp without time zone
);
ALTER TABLE ONLY users_orders ADD CONSTRAINT users_orders_pkey PRIMARY KEY (user_id, order_id);
CREATE INDEX index_users_orders_on_user_id_and_order_id ON users_orders USING btree (user_id, order_id);
```

Figure 13: Database and tables - Users Orders

```
CREATE TABLE cart_lines (
    id integer NOT NULL,
    product_id integer,
    cart_id integer,
    quantity integer DEFAULT 1,
    order_id integer,
    created_at timestamp without time zone,
    updated_at timestamp without time zone
);
CREATE SEQUENCE cart_lines_id_seq START WITH 1 INCREMENT BY 1 NO MINVALUE NO MAXVALUE CACHE 1;
ALTER SEQUENCE cart_lines_id_seq OWNED BY cart_lines.id;
ALTER TABLE ONLY cart_lines ADD CONSTRAINT cart_lines_pkey PRIMARY KEY (id);
ALTER TABLE ONLY cart_lines ALTER COLUMN id SET DEFAULT nextval('cart_lines_id_seq'::regclass);
```

Figure 14: Database and tables - Cart Lines

```
CREATE TABLE order_lines (
    id integer NOT NULL,
    product_id integer,
    order_id integer,
    created_at timestamp without time zone,
    updated_at timestamp without time zone
);
ALTER TABLE ONLY order_lines ADD CONSTRAINT order_lines_pkey PRIMARY KEY (id);
CREATE SEQUENCE order_lines_id_seq START WITH 1 INCREMENT BY 1 NO MINVALUE NO MAXVALUE CACHE 1;
ALTER TABLE ONLY order_lines ALTER COLUMN id SET DEFAULT nextval('order_lines_id_seq'::regclass);
ALTER SEQUENCE order_lines_id_seq OWNED BY order_lines.id;
```

Figure 15: Database and tables - Order Lines

```
CREATE ROLE root;
CREATE ROLE client;

GRANT INSERT ON products TO root;
GRANT DELETE ON products TO root;
GRANT UPDATE on products TO root;

REVOKE INSERT ON products FROM client;
REVOKE DELETE ON products FROM client;
REVOKE UPDATE on products FROM client;
```

Figure 16: Database and tables - Create roles on database

```
CREATE TABLE audit.logged_actions (
    schema_name text NOT NULL,
    TABLE_NAME text NOT NULL,
    user_name text,
    action_tstamp TIMESTAMP WITH TIME zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    action TEXT NOT NULL CHECK (action IN ('I','D','U')),
    original_data text,
    new_data text,
    query text
) WITH (fillfactor=100);
```

Figure 17: Database and tables - Logged Actions

```
CREATE TABLE audit.root_actions (
    schema_name text NOT NULL,
    TABLE_NAME text NOT NULL,
    user_name text,
    action_tstamp TIMESTAMP WITH TIME zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    action TEXT NOT NULL CHECK (action IN ('I','D','U')),
    original_data text,
    new_data text,
    query text
) WITH (fillfactor=100);
```

Figure 18: Database and tables - Root Actions

```
   BEGIN
      RETURN QUERY EXECUTE
           'SELECT * FROM   users WHERE  email = ''' ||
               em || ''' AND encrypted_password = crypt('''
               || passwd || ''', encrypted_password)'
      USING em;
   END
```

Figure 19: Functions - Login vulnerable

```
DECLARE
  time_now timestamp;
BEGIN
   SELECT now() into time_now;
   UPDATE users set last_sign_in_at=time_now,
                    sign_in_count=sign_in_count + 1
   WHERE id = user_id;
END
```

Figure 20: Functions - Audit login

```
DECLARE
  nr_users integer;
  user_id integer;
BEGIN
  SELECT count(users.id), max(users.id) into nr_users, user_id FROM users
  WHERE email = em AND encrypted_password = crypt(passwd, encrypted_password);

  IF nr_users <> 0 THEN
      EXECUTE audit_login(user_id);
      return 'Login success';
  ELSE
      return 'Login failed';
  END IF;
END
```

Figure 21: Functions - Login safe

```
  BEGIN
    RETURN QUERY EXECUTE
      'SELECT products.id::text || ''-->'' || products.name::text
       FROM products
       WHERE name ilike ''%' || q || '%'''
    USING q;
  END
```

Figure 22: Functions - Search product - vulnerable

```
DECLARE
prod products;
BEGIN
  SELECT array_agg(products.name) into prod
  from products where name ilike '%' || q || '%';
  return prod;
END
```

Figure 23: Functions - Search product - safe

```
DECLARE
result text;
select_statement text;
order_statement text;
BEGIN
IF(fn_name = 'avg') THEN
  select_statement = 'SELECT users.id::text || ''-->'' || avg(products.price)::text ';
  order_statement = ' ORDER BY avg(products.price) DESC';
ELSIF (fn_name = 'sum') THEN
  select_statement = 'SELECT users.id::text || ''-->'' || sum(products.price)::text ';
  order_statement = ' ORDER BY sum(products.price) DESC';
ELSIF (fn_name = 'count') THEN
  select_statement = 'SELECT users.id::text || ''-->'' || count(products.id)::text ';
  order_statement = ' ORDER BY count(products.id) DESC';
ELSE
  RAISE WARNING 'No function available';
END IF;
RETURN QUERY EXECUTE
  select_statement || 'FROM products
         JOIN order_lines on order_lines.product_id = products.id
                         JOIN users_orders on users_orders.order_id = order_lines.order_id
                         JOIN users on users.id = users_orders.user_id
                         group by users.id'
                         || order_statement;
END
```

Figure 24: Functions - Apply function

```
DECLARE
    v_old_data TEXT;
    v_new_data TEXT;
BEGIN
    IF (TG_OP = 'UPDATE') THEN
        v_old_data := ROW(OLD.*);
        v_new_data := ROW(NEW.*);
        INSERT INTO audit.logged_actions (schema_name,table_name,user_name,action,original_data,new_data,query)
        VALUES (TG_TABLE_SCHEMA::TEXT,TG_TABLE_NAME::TEXT,session_user::TEXT,substring(TG_OP,1,1),v_old_data,v_new_data, current_query());
        RETURN NEW;
    ELSIF (TG_OP = 'DELETE') THEN
        v_old_data := ROW(OLD.*);
        INSERT INTO audit.logged_actions (schema_name,table_name,user_name,action,original_data,query)
        VALUES (TG_TABLE_SCHEMA::TEXT,TG_TABLE_NAME::TEXT,session_user::TEXT,substring(TG_OP,1,1),v_old_data, current_query());
        RETURN OLD;
    ELSIF (TG_OP = 'INSERT') THEN
        v_new_data := ROW(NEW.*);
        INSERT INTO audit.logged_actions (schema_name,table_name,user_name,action,new_data,query)
        VALUES (TG_TABLE_SCHEMA::TEXT,TG_TABLE_NAME::TEXT,session_user::TEXT,substring(TG_OP,1,1),v_new_data, current_query());
        RETURN NEW;
    ELSE
        RAISE WARNING '[AUDIT.IF_MODIFIED_FUNC] - Other action occurred: %, at %',TG_OP,now();
        RETURN NULL;
    END IF;

EXCEPTION
    WHEN data_exception THEN
        RAISE WARNING '[AUDIT.IF_MODIFIED_FUNC] - UDF ERROR [DATA EXCEPTION] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
    WHEN unique_violation THEN
        RAISE WARNING '[AUDIT.IF_MODIFIED_FUNC] - UDF ERROR [UNIQUE] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
    WHEN OTHERS THEN
        RAISE WARNING '[AUDIT.IF_MODIFIED_FUNC] - UDF ERROR [OTHER] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
END;
```

Figure 25: Functions - Audit - check if data is modified

```
DECLARE
    v_old_data TEXT;
    v_new_data TEXT;
    cu TEXT;
BEGIN
    SELECT session_user::TEXT into cu;
    IF cu = 'root' THEN
        IF (TG_OP = 'DELETE') THEN
            v_old_data := ROW(OLD.*);
            INSERT INTO audit.root_actions (schema_name, table_name, user_name, action, original_data, query)
            VALUES (TG_TABLE_SCHEMA::TEXT, TG_TABLE_NAME::TEXT, session_user::TEXT, substring(TG_OP,1,1), v_old_data, current_query());
            RETURN OLD;
        END IF;
        IF(TG_OP = 'INSERT') THEN
            v_new_data := ROW(NEW.*);
            INSERT INTO audit.root_actions (schema_name,table_name,user_name,action,new_data,query)
            VALUES (TG_TABLE_SCHEMA::TEXT,TG_TABLE_NAME::TEXT,session_user::TEXT,substring(TG_OP,1,1),v_new_data, current_query());
            RETURN NEW;
        END IF;
    ELSE
        RAISE WARNING '[AUDIT.ROOT_ACTIONS] - Other action occurred: %, at %',TG_OP,now();
        RETURN NULL;
    END IF;
EXCEPTION
    WHEN data_exception THEN
        RAISE WARNING '[AUDIT.ROOT_ACTIONS] - UDF ERROR [DATA EXCEPTION] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
    WHEN unique_violation THEN
        RAISE WARNING '[AUDIT.ROOT_ACTIONS] - UDF ERROR [UNIQUE] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
    WHEN OTHERS THEN
        RAISE WARNING '[AUDIT.ROOT_ACTIONS] - UDF ERROR [OTHER] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
END;
```

Figure 26: Functions - Audit - check if root user made an action

id [PK]	email character varying	encrypted_passw character varying	reset_password_ character varyin	reset_password_sent_at timestamp without time	remember_created_at timestamp without tim	sign_in_count integer	current_sign_in_at timestamp without	last_sign_in_at timestamp without time zone
2	user2@test.com	1QlAn.7qf$....	[null]	[null]	[null]	2	[null]	2018-11-20 15:38:49.019764
1	user1@test.com	$1$7xpdqyHl$...	[null]	[null]	[null]	3	[null]	2018-11-21 01:07:21.113128

Figure 27: Data - Users

id [PK]	name character varying	description character var	price double p	image_url character varying	created_at timestamp without time zone	updated_at timestamp without time zone
1	apple watch	watch des...	3600	http://watch.jpg	2018-11-20 15:02:28.805838	2018-11-20 15:02:28.805838
2	iphone	iphone de...	2100	http://iphone.jpg	2018-11-20 15:02:28.805838	2018-11-20 15:02:28.805838
3	laptop	laptop des...	4400	http://laptop.jpg	2018-11-20 15:02:28.805838	2018-11-20 15:02:28.805838
4	drone	drone des...	1600	http://drone.jpg	2018-11-20 15:02:28.805838	2018-11-20 15:02:28.805838
5	mouse	mouse des...	150	http://mouse.jpg	2018-11-20 15:02:28.805838	2018-11-20 15:02:28.805838

Figure 28: Data - Products

id [PK]	pay_type text	delivery_method text	created_at timestamp without time zone	updated_at timestamp without time zone
1	cash	curier	2018-11-20 15:10:10.350974	2018-11-20 15:10:10.350974
2	card	posta	2018-11-20 15:10:10.350974	2018-11-20 15:10:10.350974
3	card	curier	2018-11-20 15:10:10.350974	2018-11-20 15:10:10.350974
4	cash	ridicare personala	2018-11-20 15:10:10.350974	2018-11-20 15:10:10.350974

Figure 29: Data - Orders

user_id [PK] integ	order_id [PK] intege	total_price double preci	delivery_address text	delivered boolean	created_at timestamp wit	updated_at timestamp with
1	1	35.45	strada 1	false	[null]	[null]
2	2	85	strada 2	false	[null]	[null]
3	1	925.65	strada 3	false	[null]	[null]
4	2	1585	strada 4	false	[null]	[null]

Figure 30: Data - User Orders

id [PK] in	product_id integer	order_id integer	created_at timestamp without time zone	updated_at timestamp without time zone
1	1	1	2018-11-20 15:19:54.35967	2018-11-20 15:19:54.35967
2	2	1	2018-11-20 15:20:33.699059	2018-11-20 15:21:34.525797
3	3	1	2018-11-20 15:20:33.699059	2018-11-20 15:21:34.525797
4	4	1	2018-11-20 15:20:33.699059	2018-11-20 15:21:34.525797
5	5	1	2018-11-20 15:20:33.699059	2018-11-20 15:21:34.525797

Figure 31: Data - Order Lines

schema text	table_name text	user_name text	action_tstamp timestamp with time zone	action text	original_data text	new_data text	query text
public	users	postgres	2018-11-20 13:01:58.385441+02	I	[null]	(8,user_de_...	INSERT INTO us...
public	users	postgres	2018-11-20 13:03:44.87421+02	U	(8,user_de_test...	(8,user_de_...	UPDATE users ...
public	users	postgres	2018-11-20 13:04:01.863891+02	D	(8,user_de_test...	[null]	DELETE FROM ...
public	products	postgres	2018-11-20 15:02:28.805838+02	I	[null]	(1,"apple w...	INSERT into pro...
public	products	postgres	2018-11-20 15:02:28.805838+02	I	[null]	(2,iphone,"i...	INSERT into pro...
public	products	postgres	2018-11-20 15:02:28.805838+02	I	[null]	(3,laptop,"l...	INSERT into pro...
public	products	postgres	2018-11-20 15:02:28.805838+02	I	[null]	(4,drone,"d...	INSERT into pro...
public	products	postgres	2018-11-20 15:02:28.805838+02	I	[null]	(5,mouse,"...	INSERT into pro...
public	users	postgres	2018-11-20 15:10:04.325184+02	I	[null]	(1,user1@te...	INSERT INTO us...
public	users	postgres	2018-11-20 15:10:04.325184+02	I	[null]	(2,user2@te...	INSERT INTO us...
public	orders	postgres	2018-11-20 15:10:10.350974+02	I	[null]	(1,cash,curi...	INSERT into ord...
public	orders	postgres	2018-11-20 15:10:10.350974+02	I	[null]	(2,card,post...	INSERT into ord...
public	orders	postgres	2018-11-20 15:10:10.350974+02	I	[null]	(3,card,curi...	INSERT into ord...
public	orders	postgres	2018-11-20 15:10:10.350974+02	I	[null]	(4,cash,"ridi...	INSERT into ord...
public	order_lines	postgres	2018-11-20 15:19:54.35967+02	I	[null]	(1,1,1,"2018...	INSERT into ord...
public	order_lines	postgres	2018-11-20 15:20:33.699059+02	I	[null]	(2,1,2,"2018...	INSERT into ord...
public	order_lines	postgres	2018-11-20 15:20:33.699059+02	I	[null]	(3,1,3,"2018...	INSERT into ord...
public	order_lines	postgres	2018-11-20 15:20:33.699059+02	I	[null]	(4,1,4,"2018...	INSERT into ord...

Figure 32: Data - Logged Actions

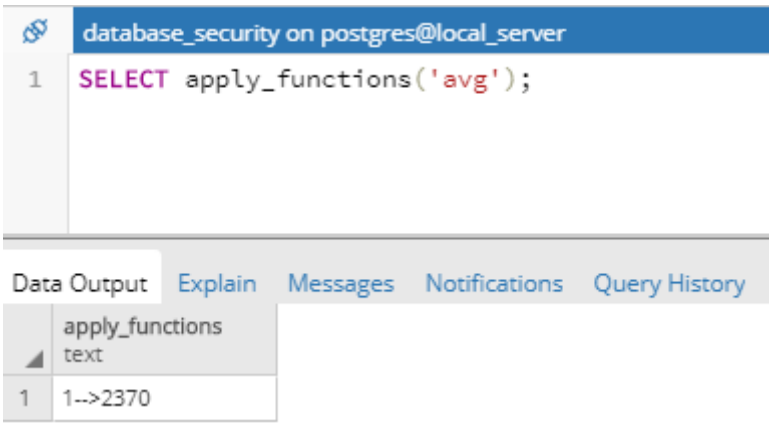

Figure 33: Experimental results - Apply function - avg

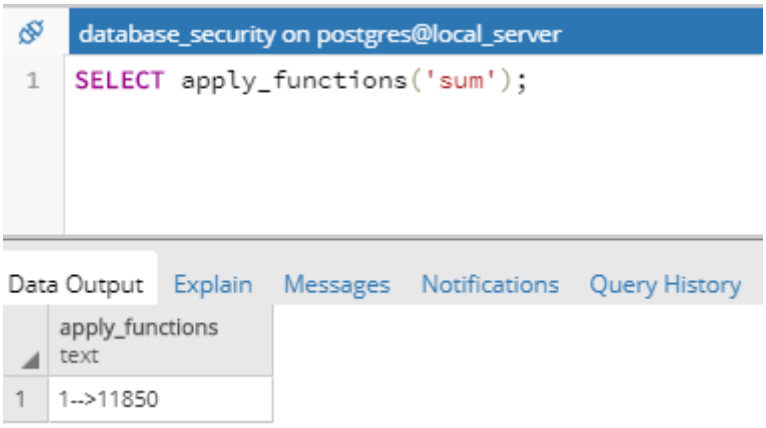

Figure 34: Experimental results - Apply function - sum

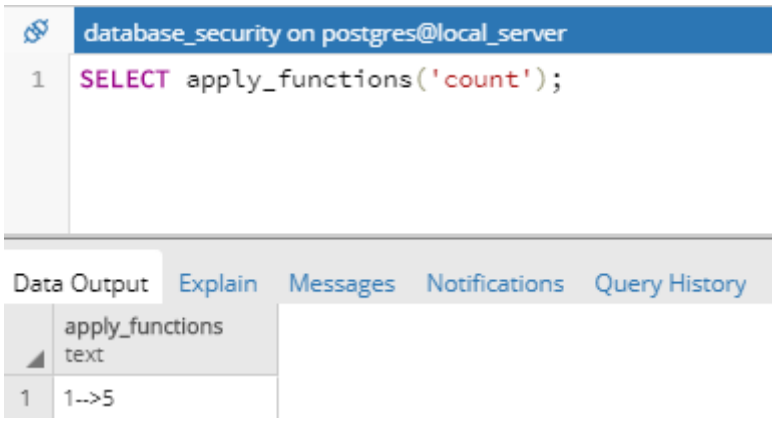

Figure 35: Experimental results - Apply function - count

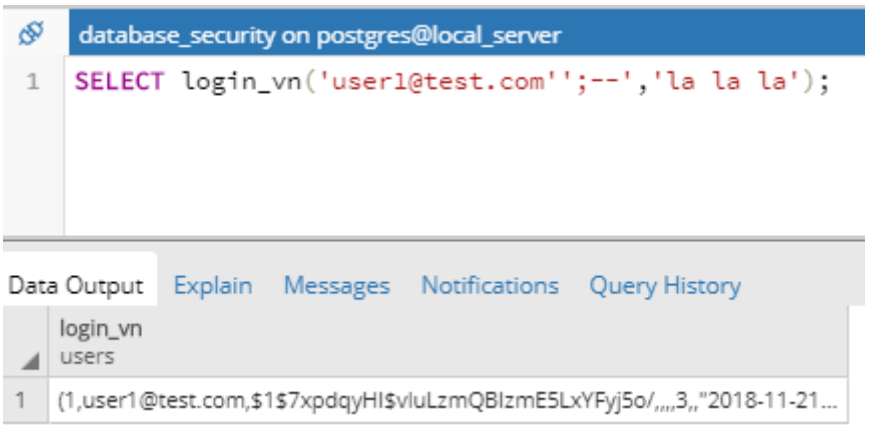

Figure 36: Experimental results - SQL INJECTION TESTS - Gain access without password

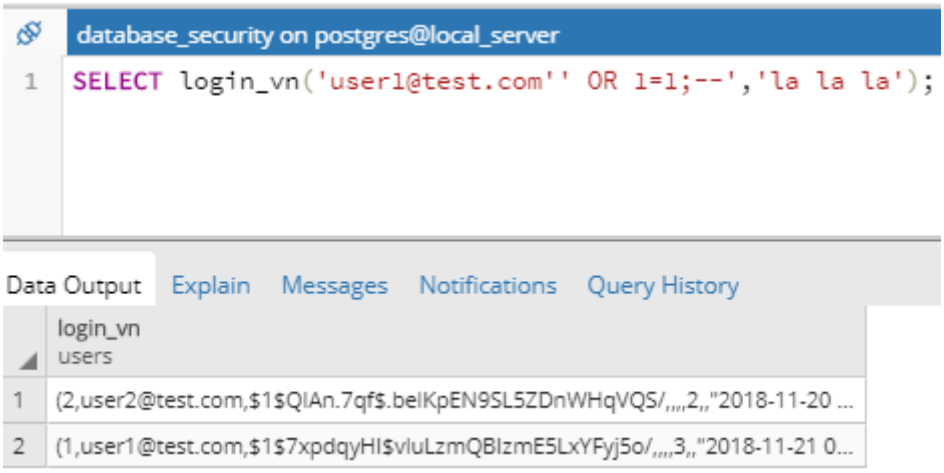

Figure 37: Experimental results - SQL INJECTION TESTS - Dump all users

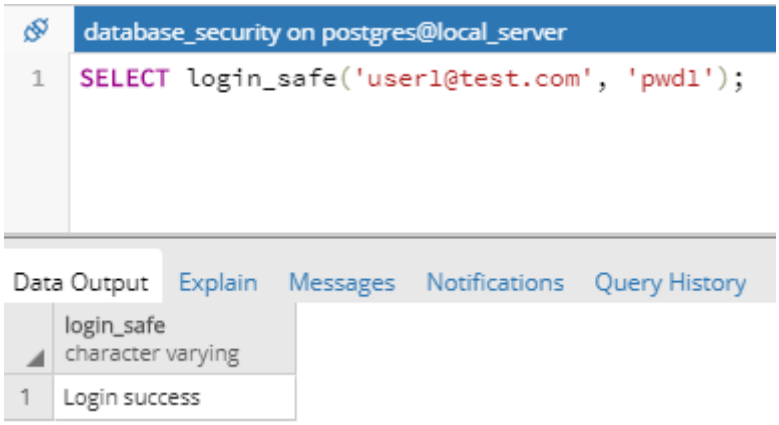

Figure 38: Experimental results - SQL INJECTION TESTS - Login secure - succes

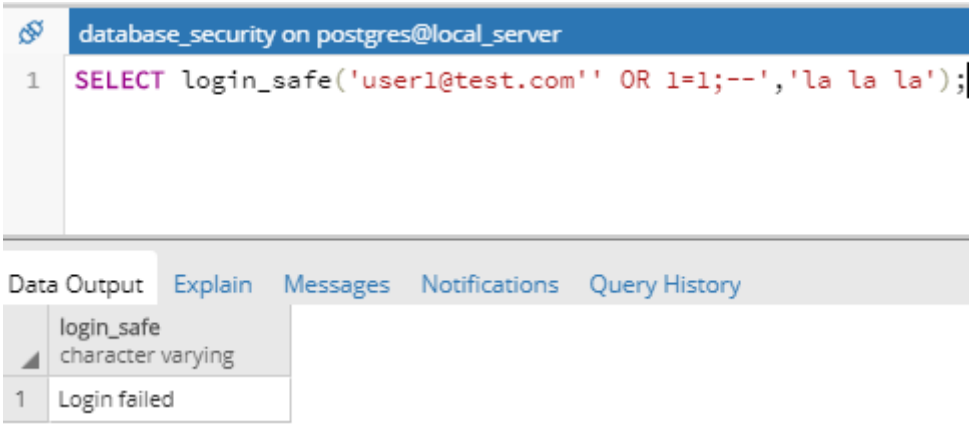

Figure 39: Experimental results - SQL INJECTION TESTS - Login secure - failed

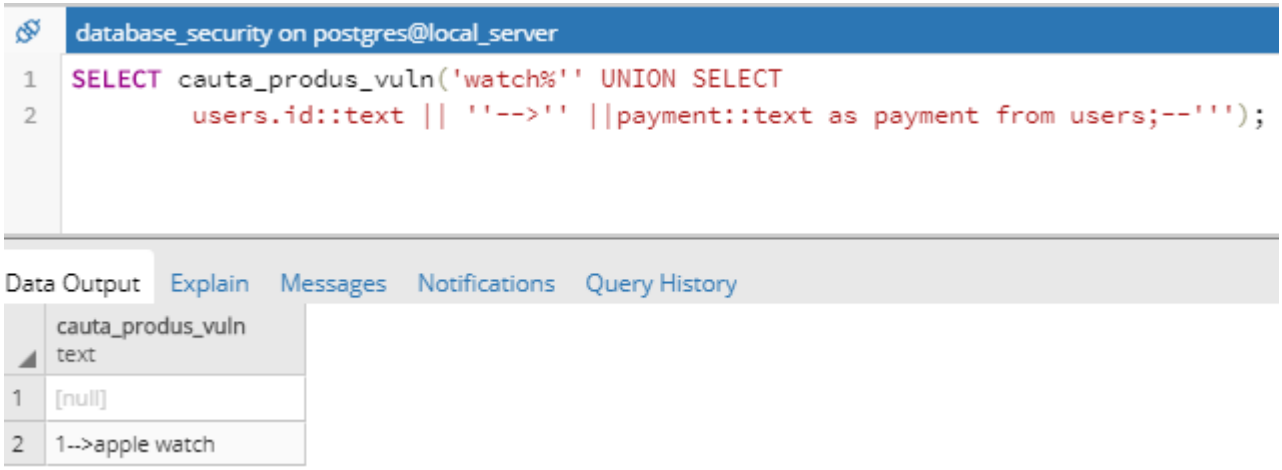

Figure 40: Experimental results - Search products - vulnerable