158.247 Database Design
Assignment 2

Preparation

In this assignment you will run a small webserver with a web page that queries a local postgres database. To begin with, create the following table in your local database `158.247` (schema public):

```
create table users(
    name            varchar(20)     not null
    , pwd           varchar(20)     not null
    , permissions   varchar(20)
    , constraint pk_users primary key (name)
);
insert into users values ('johndoe', 'pa55word', 'read');
insert into users values ('admin', 'hArd2gu3ss', 'read,write');
insert into users values ('janedoe', 'top5ecret', 'read');
```

Next create a new login role `158.247-app` with password `foobar`, and grant it access with the query:

```
CREATE USER "158.247-app" WITH PASSWORD 'foobar';
GRANT USAGE ON SCHEMA public TO "158.247-app";
GRANT SELECT ON TABLE users TO "158.247-app";
```

The program generating the page is written in python, and requires some setup.

1) If you don't have python installed yet, do that first. See https://www.python.org/downloads/. Windows users should chose python 2.7 (for ease of installing a suitable C++ compiler later), and may need to add the python directories (`C:\Python27;C:\Python27\Scripts` or similar) to their system path manually.

2) Install the python module `psycopg2` for connecting to postgres, by running one of

   ```
   > python -m pip install psycopg2-binary
   > python3 -m pip install psycopg2-binary
   ```

   Note that packages for python 2 and 3 are installed separately. You'll need psycopg2 installed for the version you are running `login.py` with, which is python 3 by default (configured at the top of `login.py`). More detailed installation instructions by OS can be found at the end of this document. A documentation of the module can be found at http://initd.org/psycopg/docs/.

On stream you will find two files available for download: `login.html` and `login.py`. Place `login.html` into a directory of your choice (in the following it will be assumed that this is your working directory). Then create a subdirectory `cgi-bin` and place `login.py` in there.

Finally you need to setup a web server to serve your script. Python already includes an HTTP server module, which can be used for this task by running one of the following commands (in the directory where `login.html` is located), for python 2 or 3 respectively:

```
> python -m CGIHTTPServer 8247
> python -m http.server --cgi 8247
> python3 -m http.server --cgi 8247
```

Python's http server will try to execute any script it finds in `cgi-bin` and send its output to port 8247. Linux and OS/X users need to ensure `login.py` has execution permission. This can be done with the command

```
> chmod a+x login.py
```

If you now point your browser at http://localhost:8247/login.html, you should see a login page. When you enter a username and password, request will be handled by login.py which checks the course database for a matching user name and password, and grants or denies access accordingly.

Tasks

1. You are to first run SQL injection attacks on the given code, then modify it to prevent them. When designing your attacks, don't assume that you know anyone's password.

   (a) Design a string that, when entered into the password field, will grant you access, regardless of what name is entered. Do not modify `login.html` or `login.py` for this. [2 marks]

   (b) Design another string that, when entered into the password field, will cause the webpage to return the admin password instead of permissions. [2 marks]
   Tip: Remember the `UNION` operator.

   (c) Modify `login.py` to prevent SQL injection attacks. [2 marks]
   Tip: Check the psycopg2 documentation.

2. Consider the tables `availability` and `appointments` below, storing doctors' availability and appointment details.

```
CREATE TABLE availability(
  doctor   varchar(20) NOT NULL
  , avl_date      date NOT NULL
  , avl_start     time NOT NULL
  , avl_end       time NOT NULL
  , CONSTRAINT pk_availability PRIMARY KEY (doctor, avl_date)
);

CREATE TABLE appointments(
    patient         varchar(20) NOT NULL
    , doctor        varchar(20) NOT NULL
    , apt_date      date        NOT NULL
    , apt_start     time        NOT NULL
    , apt_end       time        NOT NULL
    , CONSTRAINT pk_appointments PRIMARY KEY (patient, apt_date)
);
```

3. Opening hours are from 8am to 5pm each day. Add a constraint to ensure that appointment times are sensible. [2 marks]

4. Create a database function `apt_count` which takes as input a date and time and returns the number of appointments active at the given date and time. Include appointments starting at the given time. [3 marks]

5. Due to Covid-19 restrictions, the facility can allow at most 3 patients at any point in time. Create a view `fully_booked` which lists all maximal time periods (apt_date, apt_start, apt_end) during which no further appointments are possible (consider doctors' availability as well). [7 marks]
   Hint: Identify for each start time where maximal permitted appointments are reached the minimal end time at which they drop below that number again. Then eliminate non-maximal intervals.

6. Create a website which lets a user enter a day range for making an appointment, then displays a list of all time periods where the maximal number of permitted appointments has already been reached. The user should then be able to enter the details to make an appointment. When storing this appointment in the database, you must ensure that the limit of 3 concurrent appointments is not exceeded and a doctor is available as well for that time period. [12 marks]

   Feel free to use `login.html` and `login.py` as starting points, but rename them to `appointment.html` and `appointment.py` to avoid confusion when marking. For processing appointment requests, create a new file `appointment_insert.py`.

   Tip: Use `<input type="date" ...>` in `appointment.html` for entering start and end dates. To format the query result for display, you can use an html table.

Submit your source code, web page and answers (the "special password strings") via stream.
Include your name and student ID.

Detailed Setup Instructions

Linux (Ubuntu)

Python should already be installed. The module `psycopg2` requires `libpq-dev` to be installed first:

```
> sudo apt-get install libpq-dev
```

Afterwards installation proceeds using the following command:

```
> sudo python3 -m pip install psycopg2-binary
```

Note: The `login.py` script uses the `python3` interpreter by default.

Windows

For installing python and pip (which comes with python since version 2.7.9) see
http://docs.python-guide.org/en/latest/starting/install/win/

Installing the `psycopg2` module:

- Install a VisualC++ compiler for python, available at http://aka.ms/vcpython27.
- Now the module can be installed by running one of

  ```
  > python -m pip install psycopg2-binary
  > python3 -m pip install psycopg2-binary
  ```

If you're unsure where to enter these commands, search for `command line`, `command prompt` or `terminal` and educate yourself a little. The `>` symbol is not part of the commands.

OS/X

OS/X offers a nice package manager and development environment (similar to linux), but not out of the box. Hence, we begin by installing XCode, which provides compilers and libraries needed for any serious coding as well as a development environment, and homebrew, a package manager for easy installs.

- Install XCode, available here: https://developer.apple.com/xcode/download/
  Note: It's a large download (several GB). You can try to skip this step as I'm not 100% certain it's needed for the following steps, but chances are you'll want it anyway in the long run.
- Install homebrew, available here: http://brew.sh.
- Install Python – do this even if your system already has it installed, as by default essential tools such as pip (python package installer) are not included (if the command `pip` works, you can skip this step):

  ```
  > brew install python
  ```

- Install the `psycopg2` module:

  ```
  > python3 -m pip install psycopg2-binary
  ```

Troubleshooting

Below are some potential fixes for errors you might encounter:

- *Python.h: No such file or directory* – install the `python-dev` package.
- *"POST /cgi-bin/login.py HTTP/1.1" 200 - env: python3: No such file or directory* – change the first line in login.py from `#!/usr/bin/env python3` to `#!/usr/bin/env python`.
- *Symbol not found: __PyCodecInfo_GetIncrementalDecoder* – close and re-open the terminal.
- *"POST /cgi-bin/login.py HTTP/1.1" 200 -: No such file or directory* (under linux or OS/X) – you may have accidentally introduced windows line-endings (`\r`) or other funny characters. A tool like *dos2unix* can fix this.

Other common errors which probably mean you haven't been following instructions:

- *localhost refused to connect (in browser)* – you forgot to start the web server
- *"GET /login.html? HTTP/1.1" 404* – start the web server in the directory where login.html is located
- *login.py is displayed in the browser, not executed* – don't open login.html by double-clicking it (this will open it as a file), your browser bar should read `http://localhost:8247/login.html` and `http://localhost:8247/cgi-bin/login.py` respectively.
- *Import Error: No module named psycopg2* – psycopg2 hasn't been installed correctly. You can see installed packages with the command:

  ```
  > python3 -m pip list
  ```

- *psycopg2.ProgrammingError: relation "users" does not exist* – You didn't create the table or forgot to grant schema access to the newly created user