Homework 5

Never use your actual GASPAR password on COM-402 exercises (if the login form is not Tequila).

As you might notice, the solutions are available inside the docker. The goal is of course to solve the exercises without looking at the solutions.

Exercise 1: [attack] P0wn it

You can't imagine the number of bad developers out there. We just found one and we'd like you to hack his website!

Setup

You must first download the docker image containing the website alongside with the database. If you remember the lecture, you should know that having the website and database on the same server is already a sign of trouble...

Run the web server using the following command: `docker run --rm -it -p 5001:5001 --name hw5ex1 com402/hw5ex1`

Check that you can access the website through your navigator. Its IP address is given when running the previous command. Since the website is super rudimentary, here are the three URLs that you can try:

- `http://<ip>/`
- `http://<ip>/messages`
- `http://<ip>/users`

By playing around with the website you might notice that it might be vulnerable to SQL injections...

To provide you with a somehow realistic setting, we did not give you access to the webserver's code. However, for your convenience, we have provided the credentials to the webserver's *mysql* databases1:

- Access the container directly with `docker exec -it hw5ex1 /bin/bash`
- Access the database with `mysql -u<username> -p<password>`
- The credentials are stored in the `/root/credentials.cfg` file.

The solutions to the exercises can be trivially found in the database, however the point of the exercise is to write scripts that perform SQL injection attacks on the webserver to retrieve the information.

1This is clearly unrealistic, however webserver databases usually contain similar table structures and column names...

Exercise 1.1

As you might have already noticed, the website lets visitors post messages that can be then viewed on the `/messages` page. However, a secret agent with the email address *james@bond.mi5* has posted a secret message on the webserver, which for security reasons has been hidden from visitors. As you are carrying out an investigation on the activities of the government, your goal in this exercise is to find that secret message.

You must write a *python3* script that connects to the IP address of the docker in your machine and outputs the secret message in *stdout*. Your script should only use the libraries `requests` and `BeautifulSoup`. The former is a library that allows you to send organic, grass-fed HTTP/1,1 requests, without the need for manual labor. the latter is a HTML parsing library enabling you to quickly find the right information in a HTML file.

To install them :

- `pip3 install requests`
- `pip3 install beautifulsoup4`

Note : you do not need to install them if you run your script in the docker container.

To verify your solution compare the result with : `/root/solutions/secret_message.txt`

Hint : you don't always need a form for SQL injections...

Exercise 1.2

As the secret message did not contain enough information for your investigation, you searched the `/users` page and found out that a famous police inspector, *Inspector Derrick*, is registered on the website. You know that the inspector has access to criminal records that could be useful for your investigation and suspect that he used the same password to register on the vulnerable website...

Your goal is to write another script that connects to the website to retrieve his password from the database. Remember to login with the correct credentials, different from exercise 1.1.

To verify that you retrieved the correct password, compare the result with: `/root/solutions/password.txt`

Exercise 2: [defense] No SQL Injection!

You're being asked to build a super minimal website using some data stored in a MySQL database. Of course, you're starting to worry about the security nightmare this project might have.

Well not really, because so many libraries do the job for us now.

Pull and run the Docker image:

`docker run -it -p 80:80 --rm --name hw5ex2 com402/hw5ex2 bash`

You will find inside the skeleton script `site.py` where you have to fill in two endpoints `/messages` and `/users`. All information needed is included in `site.py`. The goal is to make sure your script is not susceptible to SQLi attacks. In order to modify the file `site.py`, you have to copy it to your computer, modify it and copy it back to the docker. The copy from the docker can be done with: `docker cp hw5ex2:site.py site.py`. The same command can be used to copy the file back to the docker.

Hint: check MySQL cursor.

Afterwards, execute the modified script:

```
chmod +x site.py
./site.py
```

To verify your solution, run the verify script in another bash shell inside the container:

```
docker exec -it hw5ex2 /bin/bash
```

to open a new bash shell and then:

```
./verif.py randomseed
```

If it runs and displays `GOOD`, you are done, congrats !