Parameterization and string substitution

String substitution

In the context of building a database query like so:

`CREATE TABLE fish (name TEXT, species TEXT, tank_number INTEGER)` you would likely want to include for example user input for say a search query into a database. There are two ways to go about it and the first way to build a query is via string substitution and this is the bad way.

Say the user input is in the variable `name` and the user can put whatever name they like including special characters. We have our search query to find the particular `name` that the user inputted as follows:

```
query = f"SELECT * from user_table WHERE name='{name}'"
```

Using f-string we are interpolation the variable with the query and this give the chance for SQL Injection attack to occur. This is because the user can specify something like `name = "Tom' OR 1=1"` as it's input and when the variable is used for interpolation for building the query it results into:

```
name = "Tom' OR 1 = 1"


"SELECT * from user_table WHERE name='{name}'"


"SELECT * from user_table WHERE name='Tom' OR 1 = 1"
```

And when the query is executed it will pick everyone from the database and retrieve their information (possibly private information).

Parameterized queries

Now for a much safer approach to building query is via parameterized queries. In this case, the variables that are used to build the query are pass as parameters and not used directly in a string interpolation. How you do this will depend on the library that you use, for example, in Python when

you execute a query you can pass a parameterized string along with the parameters to build a parameterized queries.

The idea is that if the user input are passed as parameters, they no longer have the chance to mess with the query since there are no interpolation, the user input is not used to build a query but rather as a parameter. **The variables themselves will no long be used as part of an executable code but rather treated as literal values.**

```
query = "SELECT * from user_table WHERE name=?"
params = (name)
cursor.execute(query, params)
```

Now no matter what the user input, even `Tom' OR 1 = 1` as `name` it will be treated literally, as you are looking for a person named `Tom' OR 1 = 1` in the database.

Further clarification

In the future, if you are back to this post and wonder isn't this the same as string substitution? You would be wrong.

The way that parameterized query works is that the query is first sent to the SQL engine, and the database will know exactly what this query will do, and only then it will insert the username and passwords as LITERAL VALUES. So the user input values cannot affect the query in anyway.

It is separating the values with the queries, the query is first sent to the database and database will know what it does, then it will ask for the parameters that you have layed out and use it LITERALLY, not as part of the query anymore.

In the previous example, you are asking the database can you select all the columns from `user_table` where the name is equal to a parameter, the database understood your request, then ask you for the `name` and the name is provided as parameter. Even if you sent `Tom' OR 1 = 1` the database will be looking for a person named `Tom' OR 1 = 1`.

Revision #2
Created 2023-01-09 18:47:00 UTC by Tamarine
Updated 2023-01-09 19:04:25 UTC by Tamarine