

Accessing Webapplication using domain name:

Ex: <https://csye6225-fall2018-nagarep.me/user/register>

The screenshot shows a REST client interface with the URL `https://csye6225-fall2018-nagarep.me/user/register` and the method `POST`. The `Authorization` tab is selected, showing `Basic Auth` with a warning about sensitive data. The `Username` field contains `harshshah1993@gmail.com` and the `Password` field contains masked characters. A `Preview Request` button is visible. The `Body` tab is selected, showing a status of `200 OK`, a time of `409 ms`, and a size of `175 B`. The response body is displayed in `JSON` format, showing a single message: `User Account has been successfully created!`.

Trying to register the same user again:

The screenshot shows the same REST client interface with the same URL and method. The `Authorization` tab is still selected. The `Body` tab is selected, showing a status of `403 Forbidden`, a time of `139 ms`, and a size of `181 B`. The response body is displayed in `JSON` format, showing a single message: `This account already exists in the system!`.

Trying to create user with .neu.edu domain ID.

https://csye6225-fall2018-nagarep.me/user/register

POST https://csye6225-fall2018-nagarep.me/user/register Send Save

Params **Authorization** Headers (2) Body Pre-request Script Tests Cookies Code

TYPE
Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Username: nagare.p@husky.neu.edu

Password: pramod

☒ Show Password

Body Cookies Headers (4) Test Results Status: 400 Bad Request Time: 155 ms Size: 193 B Download

Pretty Raw Preview JSON

1 Username has to be valid email-ID! Please try again!

Getting current time:

<https://csye6225-fall2018-nagarep.me/time>

https://csye6225-fall2018-nagarep.me/time

GET https://csye6225-fall2018-nagarep.me/time Send Save

Params **Authorization** Headers (2) Body Pre-request Script Tests Cookies Code

TYPE
Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Username: harshshah1993@gmail.com

Password:

☐ Show Password

Body Cookies Headers (4) Test Results Status: 200 OK Time: 93 ms Size: 166 B Download

Pretty Raw Preview JSON

1 Time: Mon Nov 26 20:43:56 UTC 2018

Creating Transaction:

<https://csye6225-fall2018-nagarep.me/transaction>

The screenshot shows a REST client interface with a POST request to `https://csye6225-fall2018-nagarep.me/transaction`. The request body is a JSON object: `{ "description": "ryan", "merchant": "asam", "amount": "45sm67", "date": "d458m", "category": "fffam" }`. The response status is 201 Created, with a time of 135 ms and a size of 338 B. The response body is a JSON object: `{ "id": "57b2574c-9934-463c-8ea2-15b4d99963ac384d347d-62a1-4736-890f-b81474bff6ea", "description": "ryan", "merchant": "asam", "amount": "45sm67", "date": "d458m", "category": "fffam", "attachmentsList": [] }`.

```
POST https://csye6225-fall2018-nagarep.me/transaction

{
  "description": "ryan",
  "merchant": "asam",
  "amount": "45sm67",
  "date": "d458m",
  "category": "fffam"
}
```

Status: 201 Created Time: 135 ms Size: 338 B

```
{
  "id": "57b2574c-9934-463c-8ea2-15b4d99963ac384d347d-62a1-4736-890f-b81474bff6ea",
  "description": "ryan",
  "merchant": "asam",
  "amount": "45sm67",
  "date": "d458m",
  "category": "fffam",
  "attachmentsList": []
}
```

Getting all transactions:

The screenshot shows a REST client interface with a GET request to `https://csye6225-fall2018-nagarep.me/transaction`. The request is using Basic Auth with the username `harshshah1993@gmail.com` and a password. The response status is 200 OK, with a time of 117 ms and a size of 335 B. The response body is a JSON array containing one transaction object: `[{ "id": "57b2574c-9934-463c-8ea2-15b4d99963ac384d347d-62a1-4736-890f-b81474bff6ea", "description": "ryan", "merchant": "asam", "amount": "45sm67", "date": "d458m", "category": "fffam", "attachmentsList": [] }]`.

GET https://csye6225-fall2018-nagarep.me/transaction

Status: 200 OK Time: 117 ms Size: 335 B

```
[
  {
    "id": "57b2574c-9934-463c-8ea2-15b4d99963ac384d347d-62a1-4736-890f-b81474bff6ea",
    "description": "ryan",
    "merchant": "asam",
    "amount": "45sm67",
    "date": "d458m",
    "category": "fffam",
    "attachmentsList": []
  }
]
```

Attachment for the transaction:

1. Other than png or jpeg files:

https://csye6225-fall2018-nagarep.me/transaction/57b2574c-9934-463c-8ea2-15b4d99963ac384d347d-62a1-4736-890f-b81474bff6ea/attachments

POST https://csye6225-fall2018-nagarep.me/transaction/57b2574c-9934-463c-8ea2-15b4d99963ac384d347d-62a1-4736-890f-b81474bff6ea/attachments

Params Authorization Headers (2) Body Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
file	Choose Files Document 4.docx	
Key	Value	Description

Body Cookies Headers (4) Test Results Status: 401 Unauthorized Time: 217 ms Size: 223 B Download

Pretty Raw Preview JSON

```
1 Attachment only supported for jpeg, jpg or png file extensions! Please try again!
```

2. With the supported file formats.

http://ec2-54-236-233-40.compute-1.amazonaws.com:8080/transaction/f0f6ef15-10f1-476e-96e1-eca62c7d999a27ea133f-5b99-4836-aa05-f6f626ba1153/attachments

POST http://ec2-54-236-233-40.compute-1.amazonaws.com:8080/transaction/f0f6ef15-10f1-476e-96e1-eca62c7d999a27ea133f-5b99-4836-aa05-f6f626ba1153/attachments

Params Authorization Headers (2) Body Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
file	Choose Files dummy.png	
Key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK Time: 485 ms Size: 284 B Download

Pretty Raw Preview JSON

```
1 {
2   "id": "323e74de-79bc-4fff-97b9-c1b8bd8b034d",
3   "url": "https://s3.amazonaws.com/csye6225-fall12018-nagarep.me.csye6225.com/pramodnagare1993@gmail.com_0.1902684_dummy.png"
4 }
```

3. Cross user interface:

User harshshah1993@gmail.com trying to delete transaction for user pramodnagare1993@gmail.com

https://csye6225-fall2018-nagarep.me/transaction/f0f6ef15-10f1-476e-96e1-eca62c7d999a27ea133f-5b99-4836-aa05-f6f626ba1153

DELETE ▼ https://csye6225-fall2018-nagarep.me/transaction/f0f6ef15-10f1-476e-96e1-eca62c7d999a27ea133f-5b99-48... Send Save ▼

Params **Authorization** ● Headers (2) Body ● Pre-request Script Tests Cookies Code

TYPE
Basic Auth ▼

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

! Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#) ×

Username harshshah1993@gmail.com

Password harsh

☒ Show Password

Body Cookies Headers (4) Test Results Status: 401 Unauthorized Time: 562 ms Size: 187 B Download

Pretty Raw Preview JSON ≡ 🔍

✖ 1 Oops! Something went wrong! Please try again!

4. SQL Injection:

https://csye6225-fall2018-nagarep.me/time

GET ▼ https://csye6225-fall2018-nagarep.me/time Send Save ▼

Params **Authorization** ● Headers (2) Body ● Pre-request Script Tests Cookies Code

TYPE
Basic Auth ▼

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

! Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#) ×

Username pramodnagare1993@gmail.com

Password p' OR '1' = '1';

☒ Show Password

Body Cookies Headers (4) Test Results Status: 401 Unauthorized Time: 69 ms Size: 189 B Download

Pretty Raw Preview JSON ≡ 🔍

✖ 1 No such account in the system! Please register!

Attack Vector: SQL Injection

It is a code injection technique for data driven applications in which we test the username and password authentication while in place of username or password when given 1=1(which is always true) the code which is vulnerable to attacks displays the result as when authenticated username and password is passed.

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying a SQL query that has been executed successfully. The query is as follows:

```
1 select * from user;
2
3 select * from user_transaction;
4
5 select * from user where email = '' or 1=1 ;
```

The 'Result Grid' shows the output of the query, displaying a single row of user data:

id	email	password
1	midhunvugge@gmail.com	\$2a\$11\$cAL2Gxx0XTdraoUFrvymhu7jX/hEQ/Zp...

The 'Information' tab is also visible, showing details about the 'user_transaction' table, including its columns and data types.

At the bottom of the interface, the 'Action Output' tab shows the execution details of the query, indicating that the query was executed successfully and returned 1 row(s).

Test with 'OR 1=1' field:

MySQL Workbench

aws - Warning - not supported

File Edit View Query Database Server Tools Scripting Help

Navigator: user

SCHEMAS

Filter objects

heroku_493e7650241945f

Tables

- attachments
- transactions
- user
- user_transaction

Views

Stored Procedures

Functions

Query 1: user

```
1 select * from user;
2
3 select * from user_transaction;
4
5 select * from user where email = '' or 1=1 ;
```

Result Grid

	id	email	password
1	1	midhunvugge@gmail.com	\$2a\$11\$cAL2GxxXTdraoUfrvymhu7jX/hEQ/Zp...

user 22

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:16:28	select * from user LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
2	15:16:38	select * from user where email = '' or 1=1 LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec

Table: user_transaction

Columns:

- id: varchar(40) PK
- amount: varchar(40)
- category: varchar(40)
- date: varchar(40)
- description: varchar(40)
- merchant: varchar(40)
- user_id: int(11)

Object Info

Session

Query Completed

Type here to search

3:16 PM 11/26/2018

SQL Injection helps test the vulnerability of the code. Attackers can access the data base even without knowledge of our entries. Since our web application has excessive usage of sql data base with user and user_transaction tables, it is logical and useful to test the code for SQL Injection.

5. Attachment size:

Attachment is failing, because the max limit is 1048576 Bytes or 1MB.

https://csye6225-fall2018-nagarep.me/transaction/073f5474-204c-48bb-bead-45a84f27814fdcdd4d33-d769-423b-a8e2-ba388ccba150/attachments

POST https://csye6225-fall2018-nagarep.me/transaction/073f5474-204c-48bb-bead-45a84f27814fdcdd4d33-d769-4...

Send Save

Params Authorization Headers (2) Body Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	file	<div>Choose Files</div> READ_ME.zip			
	Key	Value	Description		

Body Cookies Headers (4) Test Results Status: 500 Internal Server Error Time: 1441 ms Size: 618 B Download

Pretty Raw Preview JSON

```
1 {
2   "timestamp": "2018-11-26T22:15:06.174+0000",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "Maximum upload size exceeded; nested exception is java.lang.IllegalStateException: org.apache.tomcat.util.http.fileupload.F
6   "path": "/transaction/073f5474-204c-48bb-bead-45a84f27814fdcdd4d33-d769-423b-a8e2-ba388ccba150/attachments"
7 }
```

Attack Vector: Attachment maximum size

Attachment maximum size vulnerability test is done to verify set restrictions on the files being uploaded to our S3 bucket. In this test we pass heavy files and in a well written code, these files should not be posted as it induces considerable latency on our instance.

We are testing ‘attachment post’ in our code. An attacker can pass a large file to the S3 bucket deliberately and can induce considerable delays to our instance and can also crash the instance as well. To prevent this, we can restrict the maximum size of the files being posted into our S3 bucket.

XSS Penetration: Cross site scripting:

https://csye6225-fall2018-nagarep.me/transaction

POST

https://csye6225-fall2018-nagarep.me/transaction

Send

Save

Params

Authorization

Headers (2)

Body

Pre-request Script

Tests

Cookies

Code

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
	Authorization	Basic cHJhbW9kbmFnYXJIMTk5M0BnbWFpbC5jb20...				
<input checked="" type="checkbox"/>	Content-Type	application/json				
	Key	Value	Description			

Body

Cookies

Headers (4)

Test Results

Status: 201 Created

Time: 467 ms

Size: 354 B

Download

Pretty

Raw

Preview

JSON

1

{

2

"id": "a8ef7c2f-46b1-4bd6-9dc7-0eee44e6aef0e87a3594-f041-48be-b471-9b60e8822987",

3

"description": "<script>abc</script>",

4

"merchant": "asam",

5

"amount": "45sm67",

6

"date": "d458m",

7

"category": "fffam",

8

"attachmentsList": []

9

}