University of Helsinki
TKT200093 Cyber Security Base: Course Project I
Paula Meuronen
StudentID 012906312

Security flaws with KumpulaSalon Booking app

Table of Contents

Flaws 2-5 are based on OWASP list from 2021.

Link to the repository

https://github.com/mcpaulafi/cybersecurityproject1/

Installation and running instructions

View README.md on Git
https://github.com/mcpaulafi/cybersecurityproject1/blob/main/README.md

1 Cross-site Request Forgery

Source link(s)

[1]
https://github.com/mcpaulafi/cybersecurityproject1/blob/7378111c0aa5aeb0fd347d7e6d156bd69e8ce076/pages/views.py#L32

Description of the flaw

Attacker can book an appointment for a logged in user.

Simulation

- Start server normally with shell command **python3 manage.py runserver**
- Go to admin page http://127.0.0.1:8000/admin/
- Create at least 3 Appointments to future dates, so you have a future appointment with id 3 (or modify pages/csrf.html and start_date_id value to a id number which is in the future)

- With another shell window start another server with command **`python3 -m http.server 9000`**
- Open your Django site at **`http://127.0.0.1:8000/`** in your browser.
- Log in as a user.
- Without logging out, in the same browser tab/session, visit **`http://127.0.0.1:9000/csrf.html`**
- Appointment id 3 is now booked for user with note “CSRF attack”

How to fix the flaw

In Django every POST form requires a `{% csrf_token %}`. But CSRF protection is disabled with csrf exempt on the booking view.

pages/views.py
In the code there is following comment # SECURITY FLAW 1 [1]. Comment out @csrf_exempt

```
# SECURITY FLAW 1: CSRF
# Fix by commenting out # the line below
# @csrf_exempt
```

2 Broken Access Control

Source link(s)

[2]
https://github.com/mcpaulafi/cybersecurityproject1/blob/beaf116709caf0b818d6113abe7d19214f8a3fcf/pages/views.py#L72

Description of the flaw

Appointments view does not require authentication nor permissions. Content is available to anyone when it should be granted for a particular role (admins). Privacy violation.

Simulation

- Start server normally with shell command **python3 manage.py runserver**
- Go to admin page http://127.0.0.1:8000/
- Create some Appointments, Users and book some appointments to users
- Logout. Open your Django site at **`http://127.0.0.1:8000/appointments/`** in your browser.
- You can view all appointments and to whome they are booked without login

How to fix the flaw

pages/views.py
In the code there is following comment # SECURITY FLAW 2 [2]. Take comment # out from @staff_member_required After it is activated, only admins can view the page.

```
# SECURITY FLAW 2: BROKEN ACCESS
# Fix by removing comment # from the line below
@staff_member_required
```

3 Injection

Source link(s)

[3]
https://github.com/mcpaulafi/cybersecurityproject1/blob/3b4258ee6210c0698aa616589c72891c583e2455/pages/views.py#L58

Description of the flaw

Attacker can use form to inject malicious code.

Simulation

- Start server with shell command **python3 manage.py runserver**
- Go to admin page http://127.0.0.1:8000/admin/
- Create an Appointment in the future and a user (if you do not have any)
- Open your Django site at **`http://127.0.0.1:8000/`** in your browser.
- Log in as a user.
- Book an appointment. Fill in to the note textarea following content
  <h1 style="color: red;">Hacked page</h1>
- You get blue confirmation text: Booking successful!
- On the bottom of the page under title “Your appointments”, you can see the booked appointment with the big red note text: Hacked page

How to fix the flaw

pages/views.py
In the code there is following comment # SECURITY FLAW 3 [3]. Take comment #’s out from the if clause which checks input data.

```
# SECURITY FLAW 3: Injection
# Fix by removing comments from the if clause
    if not re.match(r'^[\w\s.,!?-]*$', note):
        messages.error(request, "Note contains invalid characters!", extra_tags="booking")
        return redirect('index')
```

4 Identification and Authentication Failures

Source link(s)

[4] Custom-made page for password change

https://github.com/mcpaulafi/cybersecurityproject1/blob/3b4258ee6210c0698aa616589c72891c583e2455/pages/urls.py#L20

[5] changepassword.html template

https://github.com/mcpaulafi/cybersecurityproject1/blob/5fb46ae6258ce41c419bd899f05b050d124a4490/pages/templates/pages/changepswd.html#L1

[6] Password quality checks

https://github.com/mcpaulafi/cybersecurityproject1/blob/3b4258ee6210c0698aa616589c72891c583e2455/pages/views.py#L206

[7] Link to password change page

https://github.com/mcpaulafi/cybersecurityproject1/blob/3b4258ee6210c0698aa616589c72891c583e2455/pages/templates/pages/index.html#L36

[9] Custom made workdlow changepasswd
https://github.com/mcpaulafi/cybersecurityproject1/blob/3b31fd85f2d97d1dc1b1d5e252619b0b17a5cfcf/pages/views.py#L187

Description of the flaw

The password change functionality allows weak or common passwords (e.g., ‘Password1’ or ‘admin/admin’), with the only restriction being a minimum length of 8 characters.

Simulation

- Start server with shell command **python3 manage.py runserver**
- Go to admin page http://127.0.0.1:8000/admin/
- Create an Appointment in the future and a user (if you do not have any)
- Open your Django site at **`http://127.0.0.1:8000/`** in your browser.
- Log in as a user.
- On Booking site click link “Change password” and go to http://127.0.0.1:8000/changepswd/
- Change your password to Password1

How to fix the flaw

There are two ways to fix this flaw.

a**) The first option** is to add more quality checks for the password on the custom made page.

pages/views.py [6]

In the code there is following comment # SECURITY FLAW 4. Take comment #’s out from the if clauses until reaching END comment.

```
# SECURITY FLAW 4: Identification and Authentication Failures:
# First option for fix is removing comments from the following lines,
# until comment END OF SECURITY FLAW
            if username == pswd1:
                messages.error(request, "Username and password should not match!",
extra_tags="pswd_check")
                return render(request, "pages/changepswd.html", context)
```

```
        if not re.search(r'\d', pswd1):
            messages.error(request, "Password must contain at least one number.",
extra_tags="pswd_check")
            return redirect('changepswd')
        if not re.search(r'[A-Z]', pswd1):
            messages.error(request, "Password must contain at least one uppercase letter.",
extra_tags="pswd_check")
            return redirect('changepswd')
        if not re.search(r'[a-z]', pswd1):
            messages.error(request, "Password must contain at least one lowercase letter.",
extra_tags="pswd_check")
            return redirect('changepswd')
        if not re.search(r'[!@#$%^&*(),.?":{}|<>]', pswd1):
            messages.error(request, "Password must contain at least one special character.",
extra_tags="pswd_check")
            return redirect('changepswd')
# END OF SECURITY FLAW
```

b) The second option, which is much preferable, is to abandon custom made system and use Djangos built-in password workflow. Just disable the custom made page and change link to the built-in page.

pages/urls.py [4]

```
# SECURITY FLAW 4: Identification and Authentication Failures
# Fix here by
# disable this line by commenting it out
# path("changepswd/", views.changepswd, name="changepswd"),
```

pages/templates/pages/index.html [7]

```
 <!--SECURITY FLAW 4: Replace the 'changepswd' with 'password_reset'-->
 <p><a href="{% url 'password_reset' %}">Change password</a></p>
```

pages/templates/pages/changepswd.html [5]

File becomes unneccessary.

pages/views.py [9]

The entire function can be deleted because it is now unused.

```
# SECURITY FLAW 4: Identification and Authentication Failures:
# Second option for fix is to disable entire function and revert to built-in workflow
def changepswd(request):
```

5 Security Misconfiguration

Source link(s)

[10] Url to answer setup page

https://github.com/mcpaulafi/cybersecurityproject1/blob/3e4e6fc9be362d1e6401e6dee239d38fc2b23ab7/pages/urls.py#L24

[11] Url to forgotten password page

https://github.com/mcpaulafi/cybersecurityproject1/blob/5fb46ae6258ce41c419bd899f05b050d124a4490/pages/urls.py#L22

[12] Question and answer setup

https://github.com/mcpaulafi/cybersecurityproject1/blob/5fb46ae6258ce41c419bd899f05b050d124a4490/pages/views.py#L85

[13] Forgotten password process

https://github.com/mcpaulafi/cybersecurityproject1/blob/5fb46ae6258ce41c419bd899f05b050d124a4490/pages/views.py#L129

[14] Link on HTML page to setting up answer

https://github.com/mcpaulafi/cybersecurityproject1/blob/5fb46ae6258ce41c419bd899f05b050d124a4490/pages/templates/pages/index.html#L34

[15] HTML page layout for forgotten password
https://github.com/mcpaulafi/cybersecurityproject1/blob/5fb46ae6258ce41c419bd899f05b050d124a4490/pages/templates/pages/forgot.html#L1

[16] HTML page layout for saving answer
https://github.com/mcpaulafi/cybersecurityproject1/blob/5fb46ae6258ce41c419bd899f05b050d124a4490/pages/templates/pages/question.html#L1

[17] Model for Question
https://github.com/mcpaulafi/cybersecurityproject1/blob/ee3a16f257e43bbd0eb7a45172f2a3ed29683297/pages/models.py#L30

[18] Model for Answer
https://github.com/mcpaulafi/cybersecurityproject1/blob/ee3a16f257e43bbd0eb7a45172f2a3ed29683297/pages/models.py#L47

[19] Link to HTML page for forgotten password
https://github.com/mcpaulafi/cybersecurityproject1/blob/ee3a16f257e43bbd0eb7a45172f2a3ed29683297/pages/templates/pages/login.html#L28

Description of the flaw

The credential recovery workflow is insecure because it relies on questions and answers, which are easy to find out or guess.

Simulation

- Start server with shell command **python3 manage.py runserver**
- Go to admin page http://127.0.0.1:8000/admin/
- Create a user

- Create a Answer. Set a question and answer linked to the user.
- Open link Forgot your password? To go to http://127.0.0.1:8000/forgot/
- Fill in the form with users information (like you were an attacker who manages to make the right guess)
- The page “Change password” opens with blue text “Recovery question was answered correctly!”
- You can now change users password

How to fix the flaw

Requires reverting to Djangos own workflow and disabling self-made systems.

1) urls.py [10]

SECURITY FLAW 5: Security Misconfiguration

Fix here by

disable these lines by commenting out

path("forgot/", views.forgot, name="forgot"),

path("question/", views.question, name="question"),

2) templates/pages/index.html [14]

<!--SECURITY FLAW 5: Remove recovery questions and answers. Remove the line below.-->
<!--p>Set recovery question</p-->

3) templates/pages/login.html [19]

<!--SECURITY FLAW 5: Replace the 'forgot' with 'password_reset'-->
Forgot your password?

4) templates/pages/question.html [16]
File becomes unneccessary.

5) pages/views.py [12] [13]

SECURITY FLAW 5: Security Misconfiguration:
Fix by disabling entire funtion and revert to built-in workflow

~~@login_required~~
~~def question(request):~~
~~[...]~~

SECURITY FLAW 5: Security Misconfiguration:
Fix by disabling entire funtion and revert to built-in workflow

~~def forgot(request):~~
~~[...]~~

6) pages/models.py [17] [18]

SECURITY FLAW 5: Security Misconfiguration:
Fix by disabling Question class
~~class Question(models.Model):~~
~~[…]~~

SECURITY FLAW 5: Security Misconfiguration:
Fix by disabling Answer class
~~class Answer(models.Model):~~
~~[…]~~