PWNIOT: Tutorial Page Documentation

By Doniyor Ismatilloev

Table of Contents

PWNIOT: Tutorial Page Documentation ***1***

Overview **2**

File Structure **2**

HTML Templates **2**

tutorial.html 2

widgets.html 2

Python Backend (Flask) **3**

tutorial.py 3

init.py 4

Usage **4**

For Administrators 4

For Users 5

Conclusion **5**

Overview

This documentation covers the implementation of the tutorial/help page in the Pwnlot web application, a platform for learning about security through Capture The Flag (CTF) challenges. The tutorial/help page allows administrators to upload educational resources in the form of PDFs and links, which are then accessible to users of the application.

File Structure

The implementation is spread across several files:

- HTML Templates:
  - dojo_theme/templates/tutorial.html: Main HTML template for the tutorial page.
  - dojo_theme/templates/macros/widgets.html: Contains macros for rendering UI components, including the newly added tutorial_card macro.
- Python Backend (Flask):
  - dojo_plugin/pages/tutorial.py: Contains Flask routes for tutorial page functionalities like viewing, uploading, and deleting tutorials.
  - dojo_plugin/__init__.py: Initialization of the Flask application, registering the tutorial blueprint.

HTML Templates

tutorial.html

- Extends Base Template: Inherits the structure from base.html.
- Content Blocks:
  - Uploaded PDFs Section: Lists PDF files uploaded by the admin.
  - Submitted Links Section: Shows links submitted by the admin.
  - Forms for Admin: If the user is an admin, forms for uploading PDFs and submitting links are displayed.

widgets.html

The tutorial_card macro is a templating tool used to generate consistent HTML structures for displaying tutorials.

- Purpose: It is designed to create a unified and styled card for each tutorial item (PDF or link). This includes displaying the title, a brief description, and a delete button for admins.
- Features:
  - Flexibility: The macro accepts parameters like title, url, text, type, and identifier, making it adaptable for different types of content.

- Admin-Specific Content: The macro includes a conditional statement to render a delete button, but only for admin users. This is aligned with the principle of showing controls relevant to a user's role.

```
{% macro tutorial_card(title, url, text, type, identifier) -%}
  <div>
    <!-- Content rendering based on passed parameters →
                <a class="text-decoration-none" href="{{ url }}" style="text-decoration: none;">
      <div class="card card-small">
        <!-- Title and Text -->
      </div>
    </a>
    <!-- Admin-specific delete button -->
    {% if is_admin() %}
      <button>Delete</button>
    {% endif %}
  </div>
{%- endmacro %}
```

- Advantages: This macro enhances code reusability and consistency across the application. It also keeps the template code clean and focused by abstracting the details of how tutorial cards are rendered.

Python Backend (Flask)

tutorial.py

- Routes:
  - /tutorial: Main route for the tutorial page.
  - /tutorials/upload-pdf: Handles PDF uploads.
  - /tutorials/submit-link: Handles link submissions.
  - /tutorials/delete/pdf/<filename>: Handles deletion of a PDF.
  - /tutorials/delete/link/<int:index>: Handles deletion of a link.
- Utility Functions:
  - allowed_file: Checks if the uploaded file is allowed (PDFs).
- Logging: Basic logging setup for error tracking.

Flask Blueprint in tutorial.py

In Flask, a Blueprint is a way to organize a group of related views and other code. Rather than registering views and other code directly with an application, they are registered with a blueprint. This blueprint is then registered with the application when it is available in the factory function.

- Blueprint Creation: In tutorial.py, a Blueprint named pwncollege_tutorial is created. This is used to group all the tutorial-related routes and functionalities.
    - tutorial = Blueprint('pwncollege_tutorial', __name__)

- Purpose: The Blueprint is used to:
- Organize tutorial-related routes in a modular fashion.
- Enhance maintainability by separating tutorial functionalities from other parts of the application.
- Registration: The Blueprint is registered to the Flask app in the __init__.py file:
    - app.register_blueprint(tutorial,url_prefix='/pwncollege_tutorial')

Cross-Site Request Forgery protection

Cross-Site Request Forgery (CSRF) protection is an essential security feature to prevent unauthorized commands from being transmitted from a user that the web application trusts. However, there are certain cases where you might need to bypass this protection. This is where @bypass_csrf_protection comes into play.

- Usage: In tutorial.py, this decorator is used in routes that handle form submissions (upload_pdf, submit_link, delete_pdf, delete_link).
- Rationale: This might be used for various reasons, such as:
    - Handling AJAX requests that might not carry the CSRF token.
    - Working with third-party services that cannot provide the token.
    - Simplifying the implementation during early development or in internal tools where security requirements are relaxed.

However, it's important to understand the security implications of bypassing CSRF protection. It should only be done after careful consideration and in scenarios where security can be ensured through other means.

init.py

- Registers the tutorial blueprint with the Flask app.
- Contains other configurations and initializations for the PWNIOT web application.

Usage

For Administrators

- Uploading PDFs:
    - Access the tutorial page.
    - Use the provided form to upload a PDF and enter its description.
- Submitting Links:
    - Access the tutorial page.
    - Use the form to submit a web link along with a description.
- Deleting Resources:

- On the tutorial page, next to each resource (PDF/link), an admin will find a delete button to remove the resource.

For Users

- Users can view the list of PDFs and links on the tutorial page.
- Resources can be accessed by clicking on their respective entries.

Conclusion

This documentation provides an overview of the tutorial/help page feature in the PwnIot website. It covers the HTML structure, Flask routes, and usage instructions for both administrators and users.