ParisNeo / lollms Public

Code Issues 6 Pull requests Actions Projects Security and quality

Commit 9767b88

ParisNeo committed on Jan 17

```
feat(social,dm): add input sanitization with bleach

- Introduce bleach-based `sanitize_content` utility
- Apply sanitization to posts, comments, DMs, and group names
- Define allowed HTML tags and attributes for safe rendering
```

main

1 parent 728d083 commit 9767b88

4 files changed +183 -7 lines changed

Top

Filter files…

- CHANGELOG.md
- backend/routers/social
  - __init__.py
  - dm.py
- scripts
  - sanitize_existing_content.py

4 files changed +183 -7 lines changed

Search within code

CHANGELOG.md

```
@@ -44,6 +44,10 @@ All notable changes to the LoLLMs Platform will be documented in this file.

## [2026-01-17 14:28]

+ - feat(social,dm): add input sanitization with bleach
```

```
48 +
49 + ## [2026-01-17 14:28]
50 +
47 51   - feat(social): add missing imports and update comments
48 52
49 53   ## [2026-01-17 12:55]
```

backend/routers/social/__init__.py

```
@@ -5,6 +5,7 @@
5  5   from fastapi import APIRouter, Depends, HTTPException, status
6  6   from ascii_colors import trace_exception
7  7   import re
   8 + import bleach
8  9   from backend.settings import settings
9  10  from backend.task_manager import task_manager
10 11  from backend.tasks.social_tasks import _respond_to_mention_task,
       _moderate_content_task
@@ -32,6 +33,32 @@
32 33  )
33 34  social_router.include_router(mentions_router, prefix="/mentions")
34 35
   36 + # --- Security: Sanitization Config ---
   37 + ALLOWED_TAGS = [
   38 +     'p', 'b', 'i', 'u', 'em', 'strong', 'a', 'br', 'ul', 'ol', 'li',
   39 +     'code', 'pre', 'blockquote', 'h1', 'h2', 'h3', 'h4', 'h5', 'h6',
   40 +     'table', 'thead', 'tbody', 'tr', 'th', 'td', 'strike', 'hr', 'span', 'div'
   41 + ]
   42 +
   43 + ALLOWED_ATTRS = {
   44 +     'a': ['href', 'title', 'target', 'rel'],
   45 +     'img': ['src', 'alt', 'title', 'width', 'height'],
   46 +     'span': ['class'],
   47 +     'div': ['class'],
   48 +     'code': ['class'],
   49 +     'pre': ['class']
   50 + }
   51 +
   52 + def sanitize_content(content: str) -> str:
```

```
+     """
+     Sanitizes user input to prevent XSS while allowing basic formatting.
+     """
+     if not content:
+         return content
+     # bleach.clean will strip or escape tags not in ALLOWED_TAGS
+     # and strip attributes not in ALLOWED_ATTRS.
+     return bleach.clean(content, tags=ALLOWED_TAGS, attributes=ALLOWED_ATTRS,
strip=True)
+
  # --- Helpers ---
  def get_post_public(db: Session, post: DBPost, current_user_id: int) ->
PostPublic:
      # Logic to convert DBPost to PostPublic, including like counts and status
@@ -149,9 +176,12 @@ def create_post(
      moderation_enabled = settings.get("ai_bot_moderation_enabled", False)
      initial_status = "pending" if moderation_enabled else "validated"

+     # Sanitize content to prevent Stored XSS
+     clean_content = sanitize_content(post_data.content)
+
      new_post = DBPost(
          author_id=current_user.id,
-         content=post_data.content,
+         content=clean_content,
          visibility=post_data.visibility,
          media=post_data.media,
          moderation_status=initial_status
@@ -162,7 +192,7 @@ def create_post(

      # --- NEW: Check for @lollms mention ---
      if settings.get("ai_bot_enabled", False):
-         if re.search(r'\B@lollms\b', post_data.content, re.IGNORECASE):
+         if re.search(r'\B@lollms\b', clean_content, re.IGNORECASE):
              task_manager.submit_task(
                  name=f"AI Bot responding to post by {current_user.username}",
                  target=_respond_to_mention_task,
@@ -241,6 +271,11 @@ def update_post(
```

```
241  271            raise HTTPException(status_code=status.HTTP_403_FORBIDDEN, detail="You can only edit your own posts.")
242  272
243  273        update_data = post_data.model_dump(exclude_unset=True)
     274  +
     275  +     # Sanitize content update
     276  +     if 'content' in update_data and update_data['content']:
     277  +         update_data['content'] = sanitize_content(update_data['content'])
     278  +
244  279        for key, value in update_data.items():
245  280            setattr(post, key, value)
246  281
     @@ -439,10 +474,13 @@ def add_comment_to_post(
439  474        moderation_enabled = settings.get("ai_bot_moderation_enabled", False)
440  475        initial_status = "pending" if moderation_enabled else "validated"
441  476
     477  +     # Sanitize comment content
     478  +     clean_content = sanitize_content(comment_data.content)
     479  +
442  480        new_comment = DBComment(
443  481            post_id=post_id,
444  482            author_id=current_user.id,
445       -          content=comment_data.content,
     483  +          content=clean_content,
446  484            moderation_status=initial_status
447  485        )
448  486        db.add(new_comment)
     @@ -452,7 +490,7 @@ def add_comment_to_post(
452  490        # Mention Response
453  491        if settings.get("ai_bot_enabled", False):
454  492            if current_user.username != 'lollms':
455       -              is_explicit_mention = re.search(r'\B@lollms\b', comment_data.content, re.IGNORECASE)
     493  +              is_explicit_mention = re.search(r'\B@lollms\b', clean_content, re.IGNORECASE)
456  494                post_author_username = post.author.username if post.author else db.query(DBUser.username).filter(DBUser.id == post.author_id).scalar()
457  495                is_bot_post = (post_author_username == 'lollms')
```

```
458  496                was_mentioned_in_post = re.search(r'\B@lollms\b', post.content,
     re.IGNORECASE)
```

backend/routers/social/dm.py

```
     @@ -11,6 +11,7 @@
11   11  from werkzeug.utils import secure_filename
12   12  from datetime import datetime
13   13  from pydantic import BaseModel, Field
     14 + import bleach
14   15
15   16  from backend.db import get_db
16   17  from backend.db.models.user import User as DBUser
     @@ -23,6 +24,26 @@
23   24
24   25  dm_router = APIRouter(prefix="/api/dm", tags=["Direct Messaging"])
25   26
     27 + # --- Security: Sanitization Config (Shared config could be moved to utils, but keeping localized for now) ---
     28 + ALLOWED_TAGS = [
     29 +     'p', 'b', 'i', 'u', 'em', 'strong', 'a', 'br', 'ul', 'ol', 'li',
     30 +     'code', 'pre', 'blockquote', 'h1', 'h2', 'h3', 'h4', 'h5', 'h6',
     31 +     'table', 'thead', 'tbody', 'tr', 'th', 'td', 'strike', 'hr', 'span', 'div'
     32 + ]
     33 + ALLOWED_ATTRS = {
     34 +     'a': ['href', 'title', 'target', 'rel'],
     35 +     'img': ['src', 'alt', 'title', 'width', 'height'],
     36 +     'span': ['class'],
     37 +     'div': ['class'],
     38 +     'code': ['class'],
     39 +     'pre': ['class']
     40 + }
     41 +
     42 + def sanitize_content(content: str) -> str:
     43 +     if not content:
     44 +         return content
     45 +     return bleach.clean(content, tags=ALLOWED_TAGS, attributes=ALLOWED_ATTRS, strip=True)
     46 +
26   47  class BroadcastDMRequest(BaseModel):
```

```
27  48          content: str = Field(..., min_length=1)
28  49
    @@ -33,6 +54,9 @@ def _broadcast_dm_task(task: Task, sender_id: int, content: str):
33  54              if not sender:
34  55                  raise Exception("Sender not found")
35  56
    57  +           # Sanitize broadcast content
    58  +           clean_content = sanitize_content(content)
    59  +
36  60              # Get all active users except sender
37  61              users = db.query(DBUser).filter(DBUser.id != sender_id, DBUser.is_active == True).all()
38  62              total = len(users)
    @@ -47,7 +71,7 @@ def _broadcast_dm_task(task: Task, sender_id: int, content: str):
47  71                  new_message = DBDirectMessage(
48  72                      sender_id=sender_id,
49  73                      receiver_id=user.id,
50      -                   content=content
    74  +               content=clean_content
51  75                  )
52  76                  db.add(new_message)
53  77                  # Commit frequently to ensure messages are saved
    @@ -89,7 +113,10 @@ async def create_group_conversation(
89  113         current_user: UserAuthDetails = Depends(get_current_active_user),
90  114         db: Session = Depends(get_db)
91  115     ):
92      -       new_conv = DBConversation(name=payload.name, is_group=1)
    116 +       # Sanitize group name
    117 +       clean_name = sanitize_content(payload.name)
    118 +
    119 +       new_conv = DBConversation(name=clean_name, is_group=1)
93  120         db.add(new_conv)
94  121         db.commit()
95  122         db.refresh(new_conv)
    @@ -198,6 +225,9 @@ async def send_direct_message(
198 225         if not receiver_user_id and not conversation_id:
```

```
199  226            raise HTTPException(status_code=400, detail="Either receiverUserId or
     conversationId must be provided.")
200  227
     228 +      # Sanitize DM content
     229 +      clean_content = sanitize_content(content)
     230 +
201  231        image_paths = []
202  232        if files:
203  233            dm_assets_path = get_user_dm_assets_path(current_user.username)
          @@ -216,7 +246,7 @@ async def send_direct_message(
216  246
217  247        new_message = DBDirectMessage(
218  248            sender_id=current_user.id,
219       -          content=content,
     249 +          content=clean_content,
220  250            image_references=image_paths if image_paths else None
221  251        )
222  252
```

scripts/sanitize_existing_content.py

```
          @@ -0,0 +1,104 @@
     1  + import sys
     2  + import os
     3  + from pathlib import Path
     4  + import bleach
     5  +
     6  + # Add the project root to python path so we can import backend modules
     7  + project_root = Path(__file__).resolve().parent.parent
     8  + sys.path.append(str(project_root))
     9  +
     10 + from backend.db import get_db, session as db_session_module
     11 + from backend.db.models.social import Post, Comment
     12 + from backend.db.models.dm import DirectMessage, Conversation
     13 +
     14 + # Configuration matching the API fix
     15 + ALLOWED_TAGS = [
     16 +     'p', 'b', 'i', 'u', 'em', 'strong', 'a', 'br', 'ul', 'ol', 'li',
     17 +     'code', 'pre', 'blockquote', 'h1', 'h2', 'h3', 'h4', 'h5', 'h6',
     18 +     'table', 'thead', 'tbody', 'tr', 'th', 'td', 'strike', 'hr', 'span', 'div'
```

```
+ ]
+
+ ALLOWED_ATTRS = {
+     'a': ['href', 'title', 'target', 'rel'],
+     'img': ['src', 'alt', 'title', 'width', 'height'],
+     'span': ['class'],
+     'div': ['class'],
+     'code': ['class'],
+     'pre': ['class']
+ }
+
+ def sanitize_content(content: str) -> str:
+     if not content:
+         return content
+     return bleach.clean(content, tags=ALLOWED_TAGS, attributes=ALLOWED_ATTRS,
  strip=True)
+
+ def run_sanitization():
+     print("--- Starting Database Sanitization ---")
+
+     # Initialize DB
+     from backend.config import APP_DB_URL
+     from backend.db import init_database
+     init_database(APP_DB_URL)
+
+     db = db_session_module.SessionLocal()
+
+     try:
+         # 1. Sanitize Posts
+         print("Scanning Posts...")
+         posts = db.query(Post).all()
+         count = 0
+         for post in posts:
+             if post.content:
+                 clean = sanitize_content(post.content)
+                 if clean != post.content:
+                     post.content = clean
+                     count += 1
+         print(f"  - Sanitized {count} Posts.")
+
```

```
+         # 2. Sanitize Comments
+         print("Scanning Comments...")
+         comments = db.query(Comment).all()
+         count = 0
+         for comment in comments:
+             if comment.content:
+                 clean = sanitize_content(comment.content)
+                 if clean != comment.content:
+                     comment.content = clean
+                     count += 1
+         print(f"  - Sanitized {count} Comments.")
+
+         # 3. Sanitize Direct Messages
+         print("Scanning Direct Messages...")
+         dms = db.query(DirectMessage).all()
+         count = 0
+         for dm in dms:
+             if dm.content:
+                 clean = sanitize_content(dm.content)
+                 if clean != dm.content:
+                     dm.content = clean
+                     count += 1
+         print(f"  - Sanitized {count} Direct Messages.")
+
+         # 4. Sanitize Group Names
+         print("Scanning Conversation Names...")
+         convs = db.query(Conversation).all()
+         count = 0
+         for conv in convs:
+             if conv.name:
+                 clean = sanitize_content(conv.name)
+                 if clean != conv.name:
+                     conv.name = clean
+                     count += 1
+         print(f"  - Sanitized {count} Conversation names.")
+
+         db.commit()
+         print("--- Sanitization Complete. Database committed. ---")
+
+     except Exception as e:
```

```
98  +         print(f"CRITICAL ERROR: {e}")
99  +         db.rollback()
100 +     finally:
101 +         db.close()
102 +
103 + if __name__ == "__main__":
104 +     run_sanitization()
```

Comments 0

Please sign in to comment.