

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.

44 44

45 45 ## [2026-01-17 14:28]

46 46

47 + - 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:

```

```

53 +     """
54 +     Sanitizes user input to prevent XSS while allowing basic formatting.
55 +     """
56 +     if not content:
57 +         return content
58 +     # bleach.clean will strip or escape tags not in ALLOWED_TAGS
59 +     # and strip attributes not in ALLOWED_ATTRS.
60 +     return bleach.clean(content, tags=ALLOWED_TAGS, attributes=ALLOWED_ATTRS,
61 +                        strip=True)

```

```

35 62 # --- Helpers ---
36 63 def get_post_public(db: Session, post: DBPost, current_user_id: int) ->
    PostPublic:
37 64     # Logic to convert DBPost to PostPublic, including like counts and status

```

↓
 ↑

@@ -149,9 +176,12 @@ def create_post(

```

149 176     moderation_enabled = settings.get("ai_bot_moderation_enabled", False)
150 177     initial_status = "pending" if moderation_enabled else "validated"
151 178

```

```

179 +     # Sanitize content to prevent Stored XSS
180 +     clean_content = sanitize_content(post_data.content)
181 +

```

```

152 182     new_post = DBPost(
153 183         author_id=current_user.id,
154 -         content=post_data.content,
184 +         content=clean_content,
155 185         visibility=post_data.visibility,
156 186         media=post_data.media,
157 187         moderation_status=initial_status

```

↕

@@ -162,7 +192,7 @@ def create_post(

```

162 192
163 193     # --- NEW: Check for @lollms mention ---
164 194     if settings.get("ai_bot_enabled", False):
165 -         if re.search(r'\B@lollms\b', post_data.content, re.IGNORECASE):
195 +         if re.search(r'\B@lollms\b', clean_content, re.IGNORECASE):
166 196             task_manager.submit_task(
167 197                 name=f"AI Bot responding to post by {current_user.username}",
168 198                 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 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 27     + # --- Security: Sanitization Config (Shared config could be moved to utils, but
28 28     +     keeping localized for now) ---
29 29     +     ALLOWED_TAGS = [
30 30     +         'p', 'b', 'i', 'u', 'em', 'strong', 'a', 'br', 'ul', 'ol', 'li',
31 31     +         'code', 'pre', 'blockquote', 'h1', 'h2', 'h3', 'h4', 'h5', 'h6',
32 32     +         'table', 'thead', 'tbody', 'tr', 'th', 'td', 'strike', 'hr', 'span', 'div'
33 33     +     ]
34 34     +     ALLOWED_ATTRS = {
35 35     +         'a': ['href', 'title', 'target', 'rel'],
36 36     +         'img': ['src', 'alt', 'title', 'width', 'height'],
37 37     +         'span': ['class'],
38 38     +         'div': ['class'],
39 39     +         'code': ['class'],
40 40     +         'pre': ['class']
41 41     +     }
42 42     +     def sanitize_content(content: str) -> str:
43 43     +         if not content:
44 44     +             return content
45 45     +         return bleach.clean(content, tags=ALLOWED_TAGS, attributes=ALLOWED_ATTRS,
46 46     +             strip=True)
26 27     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(
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'

```

```
19 + ]
20 +
21 + ALLOWED_ATTRS = {
22 +     'a': ['href', 'title', 'target', 'rel'],
23 +     'img': ['src', 'alt', 'title', 'width', 'height'],
24 +     'span': ['class'],
25 +     'div': ['class'],
26 +     'code': ['class'],
27 +     'pre': ['class']
28 + }
29 +
30 + def sanitize_content(content: str) -> str:
31 +     if not content:
32 +         return content
33 +     return bleach.clean(content, tags=ALLOWED_TAGS, attributes=ALLOWED_ATTRS,
34 +         strip=True)
35 +
36 + def run_sanitization():
37 +     print("--- Starting Database Sanitization ---")
38 +
39 +     # Initialize DB
40 +     from backend.config import APP_DB_URL
41 +     from backend.db import init_database
42 +     init_database(APP_DB_URL)
43 +
44 +     db = db_session_module.SessionLocal()
45 +
46 +     try:
47 +         # 1. Sanitize Posts
48 +         print("Scanning Posts...")
49 +         posts = db.query(Post).all()
50 +         count = 0
51 +         for post in posts:
52 +             if post.content:
53 +                 clean = sanitize_content(post.content)
54 +                 if clean != post.content:
55 +                     post.content = clean
56 +                     count += 1
57 +         print(f" - Sanitized {count} Posts.")
```

```
58 + # 2. Sanitize Comments
59 + print("Scanning Comments...")
60 + comments = db.query(Comment).all()
61 + count = 0
62 + for comment in comments:
63 +     if comment.content:
64 +         clean = sanitize_content(comment.content)
65 +         if clean != comment.content:
66 +             comment.content = clean
67 +             count += 1
68 + print(f" - Sanitized {count} Comments.")
69 +
70 + # 3. Sanitize Direct Messages
71 + print("Scanning Direct Messages...")
72 + dms = db.query(DirectMessage).all()
73 + count = 0
74 + for dm in dms:
75 +     if dm.content:
76 +         clean = sanitize_content(dm.content)
77 +         if clean != dm.content:
78 +             dm.content = clean
79 +             count += 1
80 + print(f" - Sanitized {count} Direct Messages.")
81 +
82 + # 4. Sanitize Group Names
83 + print("Scanning Conversation Names...")
84 + convs = db.query(Conversation).all()
85 + count = 0
86 + for conv in convs:
87 +     if conv.name:
88 +         clean = sanitize_content(conv.name)
89 +         if clean != conv.name:
90 +             conv.name = clean
91 +             count += 1
92 + print(f" - Sanitized {count} Conversation names.")
93 +
94 + db.commit()
95 + print("--- Sanitization Complete. Database committed. ---")
96 +
97 + 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.