

ParisNeo / lollms Public

<> Code Issues 6 Pull requests Actions Projects Security and quality

Commit 76a54f0



ParisNeo committed on Jan 1

feat(files): add SSRF protection with URL validation for image downloads

- Introduce `_validate_url` to ensure only http/https URLs pointing to public IPs are allowed.
- Prevent access to private, loopback, link-local, multicast, reserved, and metadata IP addresses.
- Resolve hostnames to IPs and apply the same restrictions.
- Integrate validation into `_download_image_to_temp` with graceful error handling.
- Update image handling in DOCX conversion to fallback to a placeholder when validation fails.

main

1 parent a6625dc commit 76a54f0

1 file changed

+53 -2

↑ Top

Filter files...

backend/routers

files.py

Search within code

backend/routers/files.py

```

@@ -7,6 +7,9 @@
7 7  import os
8 8  import tempfile
9 9  import requests
10 + import socket
11 + import ipaddress
12 + from urllib.parse import urlparse
10 13  from pathlib import Path

```

```
11 14 from typing import List, Dict, Any, Tuple, Optional
12 15
@@ -281,6 +284,42 @@ def md2_to_html(md_text: str) -> str:
281 284 ]
282 285 return markdown2.markdown(md_text, extras=extras)
283 286
287 + def _validate_url(url: str):
288 +     """
289 +     Validates a URL to prevent SSRF attacks.
290 +     Ensures the scheme is http/https and the host is not a private/local IP.
291 +     """
292 +     try:
293 +         parsed = urlparse(url)
294 +         if parsed.scheme not in ('http', 'https'):
295 +             raise ValueError(f"Invalid scheme: {parsed.scheme}")
296 +
297 +         hostname = parsed.hostname
298 +         if not hostname:
299 +             raise ValueError("Invalid hostname")
300 +
301 +         # 1. Check if hostname is an IP address
302 +         try:
303 +             ip = ipaddress.ip_address(hostname)
304 +             if ip.is_private or ip.is_loopback or ip.is_link_local or str(ip)
== "169.254.169.254":
305 +                 raise ValueError(f"Access to local/private IP {hostname} is
forbidden.")
306 +             if ip.is_multicast or ip.is_reserved:
307 +                 raise ValueError(f"Access to restricted IP {hostname} is
forbidden.")
308 +         except ValueError:
309 +             # 2. Not an IP, resolve domain to check IP
310 +             try:
311 +                 addr_info = socket.getaddrinfo(hostname, None)
312 +                 for family, _, _, _, sockaddr in addr_info:
313 +                     ip_str = sockaddr[0]
314 +                     ip = ipaddress.ip_address(ip_str)
315 +                     if ip.is_private or ip.is_loopback or ip.is_link_local or
str(ip) == "169.254.169.254":
```

```

316 +         raise ValueError(f"Domain {hostname} resolves to
private IP {ip_str}.")
317 +     except socket.gaierror:
318 +         pass # DNS resolution failed, requests will likely fail too
319 +
320 +     except Exception as e:
321 +         raise ValueError(f"URL validation failed: {str(e)}")
322 +
284 323 def _download_image_to_temp(src: str) -> str:
285 324     if src.startswith("data:"):
286 325         head, b64data = src.split(",", 1)
@@ -294,6 +333,16 @@ def _download_image_to_temp(src: str) -> str:
294 333         tf.write(data)
295 334         tf.flush(); tf.close()
296 335         return tf.name
336 +
337 +     # SECURITY: Validate URL before making request
338 +     try:
339 +         _validate_url(src)
340 +     except ValueError as e:
341 +         print(f"SSRF Protection prevented access to: {src}. Reason: {e}")
342 +         # Return a placeholder or fail gracefully?
343 +         # Creating a dummy text file or raising exception is safer
344 +         raise e
345 +
297 346     r = requests.get(src, timeout=10)
298 347     r.raise_for_status()
299 348     ext = os.path.splitext(src)[1] or ".png"
@@ -390,8 +439,10 @@ def handle_block(el):
390 439         try:
391 440             tmp = _download_image_to_temp(src)
392 441             doc.add_picture(tmp, width=Inches(5.5))
393 -         except Exception:
394 -             doc.add_paragraph(alt)
442 +         except Exception as e:
443 +             # Add text placeholder if image fails (e.g. blocked by SSRF
check)
444 +             print(f"Failed to add image to DOCX: {e}")
445 +             doc.add_paragraph(f"[Image: {alt} - Could not be loaded]")

```

```
395     446         finally:  
396     447         try:  
397     448             if tmp and os.path.exists(tmp): os.unlink(tmp)
```



Comments 0



Please [sign in](#) to comment.