_server.py

```
import logging
from http.server import SimpleHTTPRequestHandler, HTTPServer
import os
import shutil
import re
import json
import requests
import feedparser

# Create a custom logger
logger = logging.getLogger('server_logger')
logger.setLevel(logging.INFO)

# Create handler
ch = logging.StreamHandler()
ch.setLevel(logging.INFO)

# Create formatter and add it to the handler
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
%(message)s')
ch.setFormatter(formatter)

# Add handler to the logger
logger.addHandler(ch)

class CustomHTTPRequestHandler(SimpleHTTPRequestHandler):
```

```
def do_POST(self):
    content_length = int(self.headers['Content-Length'])
    content_type = self.headers['Content-Type']

    # Extract boundary from content type
    boundary = content_type.split("boundary=")[1].encode()
    body = self.rfile.read(content_length)

    try:
        parts = self.parse_multipart(body, boundary)
        for part in parts:
            if 'filename' in part['headers']['Content-Disposition']:
                filename = part['headers']['Content-
Disposition'].split('filename=')[1].strip('"')
                sanitized_filename = self.sanitize_filename(filename)

                if sanitized_filename == 'fake_passwd':
                    self.send_response(200)
                    self.send_header('Content-type', 'text/plain')
                    self.end_headers()
                    self.wfile.write(b"root:x:0:0:root:/root:/bin/bash\n")
                    return

                upload_path = os.path.join('uploads', sanitized_filename)
                self.ensure_directory('uploads')

                with open(upload_path, 'wb') as f:
                    f.write(part['body'])

                self.move_file(upload_path)
            self.send_response(200)
            self.end_headers()
            self.wfile.write(b"File uploaded successfully")
            return

        self.send_response(400)
        self.end_headers()
        self.wfile.write(b"File upload failed")
    except Exception as e:
        logger.error(f"Error during file upload: {e}")
        self.send_response(500)
        self.end_headers()
        self.wfile.write(b"Internal server error")

def parse_multipart(self, body, boundary):
    parts = []
    boundary = b'--' + boundary
    for part in body.split(boundary):
        if part and part != b'--\r\n':
            part = part.strip(b'\r\n')
            try:
                headers, body = part.split(b'\r\n\r\n', 1)
                headers = self.parse_headers(headers.decode())
                parts.append({'headers': headers, 'body': body})
            except ValueError as ve:
                logger.error(f"Error parsing part: {ve}")
                logger.debug(f"Part content: {part[:100]}...")
    return parts
```

```
def parse_headers(self, headers):
    header_dict = {}
    for line in headers.split('\r\n'):
        try:
            key, value = line.split(': ', 1)
            header_dict[key] = value
        except ValueError:
            logger.error(f"Error parsing header line: {line}")
    return header_dict

def do_GET(self):
    if self.path == '/upload':
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(self.upload_form().encode())
    elif self.path == '/fetch_rss':
        self.send_response(200)
        self.send_header('Content-type', 'application/json')
        self.end_headers()
        rss_content = self.fetch_rss_feed()
        self.wfile.write(json.dumps(rss_content).encode())
    else:
        super().do_GET()

def upload_form(self):
    return '''<html>
            <body>
            <form enctype="multipart/form-data" method="post" action="/upload">
              <input type="file" name="file" />
              <input type="submit" value="Upload" />
            </form>
            </body>
            </html>'''

    def move_file(self, filepath):
        """Move files based on their extensions or prefixes."""
        if filepath.endswith(('.mp3', '.wav')):
            self.move_music_files(filepath)
        elif os.path.basename(filepath).startswith('scripts_'):
            self.move_script_files(filepath)
        elif os.path.basename(filepath).startswith('cyberoperations_'):
            self.move_cyberoperations_files(filepath)

    def move_music_files(self, filepath):
        """Move .mp3 and .wav files to the Music-files directory."""
        self.ensure_directory('Music-files')
        shutil.move(filepath, os.path.join('Music-files', os.path.basename(filepath)))

    def move_script_files(self, filepath):
        """Move files with 'scripts_' prefix to the scripts directory."""
        self.ensure_directory('scripts')
        shutil.move(filepath, os.path.join('scripts', os.path.basename(filepath)))

    def move_cyberoperations_files(self, filepath):
        """Move files with 'cyberoperations_' prefix to the CyberOperations directory."""
        self.ensure_directory('CyberOperations')
        shutil.move(filepath, os.path.join('CyberOperations',
os.path.basename(filepath)))

    def ensure_directory(self, directory):
        """Ensure that a directory exists; if not, create it."""
        if not os.path.exists(directory):
            os.makedirs(directory)

    def sanitize_filename(self, filename):
        """Sanitize the filename to avoid directory traversal attacks."""
        filename = os.path.basename(filename)  # Ensure we're only working with the
filename
        sanitized = re.sub(r'[^a-zA-Z0-9._-]', '_', filename)
        # Detect directory traversal attempts
        if '..' in filename or filename.startswith('/'):
            return 'fake_passwd'
        return sanitized

    def fetch_rss_feed(self):
        url = "https://feeds.feedburner.com/securityweek"
        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36'}
        response = requests.get(url, headers=headers)
        feed = feedparser.parse(response.content)
        rss_content = []
        for entry in feed.entries:
            rss_content.append({
                "title": entry.title,
                "link": entry.link,
                "published": entry.published
        })
      return rss_content


def run(server_class=HTTPServer, handler_class=CustomHTTPRequestHandler,
port=8000):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    logger.info(f'Starting httpd on port {port}...')
    httpd.serve_forever()


if __name__ == '__main__':
    run()
```