CWE-22 Path Traversal in load_template() — .jinja/.yaml/.yml file read without authentication

Moderate **oobabooga** published **GHSA-85fx-vw25-4c95** 5 days ago

Package

text-generation-webui (pip)

Affected versions	Patched versions
< 4.3	4.3

Description

Summary

An unauthenticated path traversal vulnerability in `load_template()` allows reading files with `.jinja`, `.jinja2`, `.yaml`, or `.yml` extensions from anywhere on the server filesystem.
For `.jinja` files the content is returned verbatim; for `.yaml` files a parsed key is extracted.

Details

The vulnerable code is in `modules/training.py` at lines 228-237:

```
def load_template(name):
    path = shared.user_data_dir / 'instruction-templates'
    for ext in ['jinja', 'jinja2', 'yaml', 'yml']:
        filepath = path / f'{name}.{ext}'
        if filepath.exists():
            if ext in ['jinja', 'jinja2']:
                return filepath.read_text(encoding='utf-8')
            else:
                data = yaml.safe_load(filepath.read_text(encoding='utf-8'))
                return data.get('instruction_template', '')
```

The `name` parameter comes from a Gradio Dropdown and is not sanitized.
Notably, `clean_path()` exists in the same file at line 190 (replaces `..` with `_`) and is applied to `lora_name` in `do_train()`, but it is not applied to `name` in `load_template()`.
The four-extension iteration broadens the attack surface compared to the other path traversal findings.

PoC

1. Clone the repository and start the server.
2. Send a crafted API request with a traversal payload as the template name.
3. The server iterates over four extensions, opens the first matching file, and returns its content.

I verified this by cloning the repository, running the verbatim `load_template()` function with traversal payloads, and confirming both `.yaml` parsed content and `.jinja` verbatim content leakage from outside the instruction-templates directory.

poc.zip

Impact

Any `.jinja`, `.jinja2`, `.yaml`, or `.yml` file readable by the server process can be exfiltrated.
No authentication required by default.
The `clean_path()` fix already exists in the codebase but is not applied to this function.

Remediation: apply `os.path.basename(name)` or the existing `clean_path()` before path construction.

We believe this qualifies as a valid security issue.
If you agree, we'd appreciate the following credit on the CVE:
Reported by Woohyun Choi, Sunwoo Lee, and Seunghyun Yoon (Korea Institute of Energy Technology, KENTECH)

Severity

Moderate 5.3 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-35483

Weaknesses

▶ CWE-22

Credits

programsurf	Reporter
woohyunchoi-kentech	Reporter
yoonsh	Reporter