

AnalogyC0de / public_exp Public[Code](#) [Issues 10](#) [Pull requests](#) [Actions](#) [Projects](#) [Security and quality](#)

New issue



Stored SSRF in QingdaoU Onlinejudge Judge Server service_url #27

Open

AnalogyC0de opened 3 weeks ago

Owner ...

Stored SSRF in Onlinejudge Judge Server service_url

Identification

- **Project:** OnlineJudge
- **Repository:** <https://github.com/QingdaoU/OnlineJudge>
- **Affected Version/Commit:** <=v1.6.1

CVE Description

A stored Server-Side Request Forgery (SSRF) vulnerability exists in the Judge Server dispatcher of QingdaoU OnlineJudge. An attacker with access to a judge server token can submit a malicious `service_url` via the `/api/admin/judge_server_heartbeat` endpoint, which is then stored in the database. When the application subsequently processes judge tasks, it uses this unvalidated URL to construct and send internal HTTP requests. This allows the attacker to force the server to make arbitrary requests to internal network resources, potentially leading to metadata exfiltration, internal network scanning, or remote code execution.

Affected Component

- **File(s):** `conf/views.py`, `judge/dispatcher.py`, `models.py`
- **Function / Method:** `ChooseJudgeServer.__enter__()`, `requests.post()`
- **Entry Point:** `POST /api/admin/judge_server_heartbeat` (`service_url` parameter)

Reproduction Summary

1. An attacker with a judge server token sends a `POST` request to `/api/admin/judge_server_heartbeat` providing a malicious internal URL (e.g., cloud metadata endpoint) in the `service_url` parameter.
2. The application stores the malicious `service_url` in the `JudgeServer` MySQL table without performing validation.
3. The judge dispatcher picks up a task and prepares to send it to the judge server.
4. The application calls `requests.post(urljoin(server.service_url, "/judge"), data=data)`, triggering an SSRF attack against the internal network.

Technical Details

- Entry Point: `/api/admin/judge_server_heartbeat` (`conf/views.py:132-163`)
- Storage: `JudgeServer.service_url` (MySQL database)
- Consumption: `judge/dispatcher.py:59-64, 155-161`
- Sink: `requests.post(urljoin(server.service_url, "/judge"), data=data)`



Validation Notes

- ✓ Entry Point: `POST /api/admin/judge_server_heartbeat` (`views.py:132-149`)
- User controllability proven: `request.data["service_url"]` → `server.service_url`
- Authentication: Token-based
- Validation: None (only `max_length=256` in serializer)

- ✓ Storage: `JudgeServer.service_url` (MySQL `judge_server` table, `models.py:15`)
- Data persists as `TextField(null=True)`
- No transformation of stored URL

- ✓ Consumption & Sink: `requests.post(urljoin(server.service_url, "/judge"), ...)` (`dispatcher.py:161` → `64`)
- Read operation: `ChooseJudgeServer.__enter__()` (`dispatcher.py:38-48`)
- Sink type: Correct - `requests.post()` is genuine SSRF sink
- No re-validation: Database value used directly

- ✓ Complete Chain: Entry → Storage → Read → Sink proven



[Sign up for free](#) to join this conversation on [GitHub](#). Already have an account? [Sign in to comment](#)

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

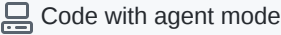

Milestone

No milestone

Relationships

None yet

Development

 Code with agent mode 

No branches or pull requests

Participants

