


jeecgboot / JeecgBoot Public[Code](#) [Issues 9](#) [Pull requests 2](#) [Actions](#) [Projects](#) [Security and quality](#)[New issue](#)

# [Security] Remote Code Execution via Unsafe Reflection in jeecgboot\_JeecgBoot #9552

[Closed](#)

Labels

[bug](#) AnalogyC0de opened 2 weeks ago ...

## Remote Code Execution via Unsafe Reflection in jeecgboot\_JeecgBoot

### Identification

- **Affected Version:** <= v3.9.1

### CVE Description

A second-order Remote Code Execution (RCE) vulnerability exists in jeecgboot\_JeecgBoot due to unsafe reflection within the `FillRuleUtil` component. The `/sys/fillRule/edit` endpoint lacks role-based authorization and input validation, allowing any authenticated user to modify the `ruleClass` parameter of an existing fill rule (e.g., `org_num_role`) and store a malicious class name in the database. When an administrator or user subsequently creates a department via the `/sys/sysDepart/add` endpoint, the application retrieves the poisoned rule and instantiates the malicious class using `Class.forName().newInstance()`, resulting in arbitrary code execution.

### Affected Component

- **File(s):** `SysFillRuleController.java` (L96-100), `SysDepartController.java` (L213-230), `SysDepartServiceImpl.java` (L187-227), `FillRuleUtil.java` (L30-84)
- **Function / Method:** `SysFillRuleController.edit()`, `FillRuleUtil.executeRule()`

- **Entry Point:** `POST /sys/fillRule/edit` (Injection), `POST /sys/sysDepart/add` (Trigger)

## Reproduction Summary

1. Identify the ID of an existing fill rule (e.g., `org_num_role`) via `GET /sys/fillRule/list`.
2. Send a `POST` request to `/sys/fillRule/edit` to inject a malicious class (e.g., `org.apache.commons.collections3.functors.InvokerTransformer`) into the `ruleClass` field.
3. Trigger the execution by sending a `POST` request to `/sys/sysDepart/add` to create a new department.
4. The application reads the injected payload from the database and executes arbitrary code upon reflecting the malicious class.

## Technical Details

### Phase 1: Infection (T0) - Write to Database ( `SysFillRuleController.java:96-100` )

The endpoint lacks a `@RequiresPermissions` check and fails to validate the `ruleClass` field:

```
@RequestMapping(value = "/edit", method = {RequestMethod.PUT, RequestMethod.POST})
public Result<?> edit(@RequestBody SysFillRule sysFillRule) {
    sysFillRuleService.updateById(sysFillRule);
    return Result.ok("编辑成功!");
}
```



### Malicious Payload for Phase 1:

```
{
  "id": "<existing_fill_rule_id>",
  "ruleCode": "org_num_role",
  "ruleClass": "org.apache.commons.collections3.functors.InvokerTransformer",
  "ruleParams": "{\"methodName\": \"exec\", \"args\": [\"calc.exe\"]}"
}
```



### Phase 3: Detonation (T1) - Sink ( `FillRuleUtil.java:36-77` )

When a department is created, `FillRuleUtil.executeRule()` fetches the poisoned rule and insecurely instantiates it:

```
// Line 36-38: Query sys_fill_rule table
QueryWrapper queryWrapper = new QueryWrapper();
queryWrapper.eq("rule_code", ruleCode); // "org_num_role"
JSONObject entity = JSON.parseObject(JSON.toJSONString(impl.getOne(queryWrapper)));

// Line 44: Extract ruleClass from database (now attacker-controlled)
String ruleClass = entity.getString("ruleClass");

// Line 77: ⚠ UNSAFE REFLECTION - RCE HERE ⚠
```



```
IFillRuleHandler ruleHandler = (IFillRuleHandler) Class.forName(ruleClass).newInstance();  
return ruleHandler.execute(params, formData);
```

## Validation Notes

### Exploit Chain Summary:

T0: Attacker (any authenticated user)  
POST /sys/fillRule/edit  
→ Stores malicious ruleClass in sys\_fill\_rule table

Dormancy: Malicious config waits in database

T1: Admin (normal department creation)  
POST /sys/sysDepart/add  
→ SysDepartServiceImpl.saveDepartData()  
→ FillRuleUtil.executeRule("org\_num\_role", ...)  
→ Queries sys\_fill\_rule table  
→ Extracts attacker-controlled ruleClass  
→ Class.forName(maliciousClass).newInstance()  
→ RCE!



### Clarifications:

- **Authorization:** The vulnerability requires standard authentication to exploit since the endpoint is protected from unauthenticated access, but it completely lacks proper role-based authorization (no `@RequiresPermissions`), allowing any standard user to overwrite the rule.
- **Dependencies:** Exploitability assumes a suitable gadget chain (like Apache Commons Collections) is present on the classpath.



**AnalogyC0de** added **bug** 2 weeks ago



zhangdaiscott 2 weeks ago

Member ...

已修复，下个版本发布



**zhangdaiscott** closed this as completed 2 weeks ago



shadowsock5 2 weeks ago

...

不是有强制类型转换吗？

```
IFillRuleHandler ruleHandler = (IFillRuleHandler) Class.forName(ruleClass).newInstance()
```



能成功吗？



AnalogyC0de 2 weeks ago

Author



已确认当前条件已经过滤无法绕过

Sign up for free

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

### Metadata

#### Assignees

No one assigned

#### Labels

bug

#### Type

No type

#### Projects

No projects

#### Milestone

No milestone

#### Relationships

None yet

#### Development

No branches or pull requests

#### Participants



