Restrizione Type Reference in SpEL

https://icarocomix.github.io/appuntidiprogrammazione

Restrizione Type Reference in SpEL

ANALISI TECNICA

PROBLEMA

Un utente malintenzionato potrebbe inserire un'espressione nel database che, una volta renderizzata, esegue codice sulla JVM.

PERCHÉ

Sandbox dell'engine SpEL. Ho scelto di limitare le classi risolvibili dal compilatore SpEL per impedire l'invocazione di metodi di sistema pericolosi.

ESEMPIO IMPLEMENTATIVO - PART 1

```
/* Implemento un TypeLocator restrittivo che blocca l'accesso
    alle classi pericolose. Il TypeLocator è il componente SpEL
    responsabile di risolvere T(java.lang.Runtime): sostituendolo
     con uno personalizzato, posso bloccare selettivamente le
    classi. */
public class RestrictedTypeLocator implements TypeLocator {
    private static final Set
<String>
    BLOCKED_PACKAGES = Set.of( "java.lang.Runtime",
        "java.lang.ProcessBuilder", "java.lang.Process",
        "java.lang.reflect", "java.lang.ClassLoader",
        "sun.misc.Unsafe", "java.io.File", "java.nio.file",
        "java.net.URL", "java.net.Socket" ); @Override public
        Class
    <?>
        findType(String typeName) throws EvaluationException {
            // Blocco l'accesso a qualsiasi classe nei package
                pericolosi for (String blocked :
                BLOCKED_PACKAGES) { if
                (typeName.startsWith(blocked)) { throw new
                EvaluationException( "Accesso negato alla classe:
                 " + typeName + ". Tipo non consentito nel
                contesto sandbox."); } } try { return
                ClassUtils.forName(typeName,
                ClassUtils.getDefaultClassLoader()); } catch
                (ClassNotFoundException e) { throw new
                EvaluationException("Classe non trovata: " +
                typeName); } } }
            /* Configuro il contesto di valutazione SpEL sandbox
                per il motore del CMS: */
```

ESEMPIO IMPLEMENTATIVO - PART 2

```
            @Configuration public class SpelSandboxConfig { @Bean
                 public SpelSandboxEvaluator
                spelSandboxEvaluator() { return new
                SpelSandboxEvaluator(); } }

@Service public
                class SpelSandboxEvaluator { private final
                SpelExpressionParser parser = new
                SpelExpressionParser(); public Object
                evaluate(String expression, Map
            <String, Object>
                variables) { StandardEvaluationContext context =
                    new StandardEvaluationContext();
                // Sostituisco il TypeLocator di default con
                    quello restrittivo context.setTypeLocator(new
                     RestrictedTypeLocator()); // Imposto solo le
                     variabili esplicitamente consentite: zero
                    bean Spring accessibili
                    variables.forEach(context::setVariable); //
                    NON chiamo context.setBeanResolver():
                    impedisco l'accesso ai bean Spring try {
                    Expression expr =
                    parser.parseExpression(expression); return
                    expr.getValue(context); } catch
                    (EvaluationException e) {
                    log.warn("Espressione SpEL bloccata: '{}' -
                    Motivo: {}", expression, e.getMessage());
                    throw new SecurityException("Espressione non
                    consentita: " + e.getMessage()); } } }
                /* Uso il valutatore sandbox nel servizio del
```

```
    CMS: */
@Service public class CmsTemplateService {
    @Autowired private SpelSandboxEvaluator
    sandbox; public String
    renderCmsBlock(CmsBlock block, PageContext
    pageContext) {
    // Valuto le espressioni SpEL presenti nel
        template CMS in modo sicuro Map<String,
        Object> safeVariables = Map.of( "page",
        pageContext.getPageData(), // Solo i dati
         della pagina "user",
        pageContext.getPublicUserData() // Solo i
         dati pubblici dell'utente // NON
        espongo: applicationContext, environment,
         system properties ); // Processo le
        espressioni ${...} nel contenuto del
        blocco CMS return
        processExpressions(block.getContent(),
        safeVariables); } private String
        processExpressions(String content,
        Map<String, Object> vars) { // Pattern
        per trovare le espressioni ${...} nel
        contenuto del CMS return
        content.replaceAll("\\$\\{([^}]+)}",
        match -> { String expression =
        match.replaceAll("\\$\\{|}", ""); try {
        Object result =
        sandbox.evaluate(expression, vars);
        return result != null ? result.toString()
         : ""; } catch (SecurityException e) {
```

ESEMPIO IMPLEMENTATIVO - PART 4

```
return "[Espressione non consentita]"; //
 Non espongo l'errore all'utente finale }
 }); } }
```

CONSOLE DI DEBUG

```
> Stato post: [utile]

> Azione: salva_post_ora()


> Requisito:
  Click sull'icona Segnalibro
```